Practical first order methods for large scale semidefinite programming

Stephen Tu * Jingyan Wang †

December 17, 2014

This paper investigates first order methods for solving large scale semidefinite programs. While interior point methods are (a) theoretically sound and (b) effective and robust in practice, they are only practical for small scale problems. As the dimension of the problem increases, both the space and time needed become prohibitive. We survey first order methods which have been proposed in the literature to deal with this issue, such as the low-rank factorization method of [BM03b], the block coordinate descent method of [WGMS09], the dual ascent approach proposed in the context of spectral bundle methods [HR00], and the recent eigenvalue saddle point transformation proposed by [Ren14]. These methods are all characterized by simple operations per iteration: matrix-vector multiplications, vector dot products, and top eigenvalue-eigenvector pair computations. These operations can exploit the sparse structure of the original problem and can be efficiently implemented using standard numerical libraries. We conclude our survey with an experimental evaluation on various max-cut and matrix completion problems.

*EECS Department, University of California, Berkeley. stephent@berkeley.edu
†EECS Department, University of California, Berkeley. jingyanw@berkeley.edu

Contents

1 Introduction

Notation. Throughout this paper, we will use the following notation. Let $\mathcal{S}^n$ denote the space of all symmetric $n \times n$ matrices. Given $X, Y \in \mathcal{S}^n$, let $\langle X, Y \rangle$ denote the Frobenius inner product

$$\langle X, Y \rangle = \mathbf{tr}(X^T Y)$$

Given an $X \in \mathcal{S}^n$, we write $X \succeq 0$ (resp. $X \succ 0$) to denote that X is positive semi-definite (resp. positive definite). We also write $\lambda_1(X) \geq \lambda_2(X) \geq ... \geq \lambda_n(X)$ to denote the eigenvalues of X in decreasing order. Given an arbitrary $X \in \mathbb{R}^{m,n}$, we write $\sigma_1(X) \geq \sigma_2(X) \geq ... \geq \sigma_{\min\{m,n\}}(X) \geq 0$ to denote the singular values of X in decreasing order.

Given an $x \in \mathbb{R}^n$, we use $\|x\|$ to denote its l_2 Euclidean norm. Similarly, given an $X \in \mathbb{R}^{m,n}$, we use $\|X\|_F$ to denote its Frobenius norm.

1.1 Semidefinite programming

In this paper, we study the following optimization problem

$$\min_{X \in \mathcal{S}^n} \langle C, X \rangle : \mathcal{A}(X) = b, X \succeq 0 \tag{1}$$

where $b \in \mathbb{R}^m$, $C \in \mathcal{S}^n$, and $\mathcal{A} : \mathcal{S}^n \longrightarrow \mathbb{R}^m$ is a linear operator of the form

$$\mathcal{A}(X) = \begin{bmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{bmatrix}$$

with $A_i \in \mathcal{S}^n$, $i = 1, ..., m$. Denote by $\mathcal{A}^* : \mathbb{R}^m \longrightarrow \mathcal{S}^n$ the adjoint operator, which has the form

$$\mathcal{A}^*(y) = \sum_{i=1}^m y_i A_i$$

A simple manipulation yields that the dual of (1) is given as

$$\max_{y \in \mathbb{R}^m} -b^T y : C + \mathcal{A}^*(y) \succeq 0 \tag{2}$$

A standard result states that if both the primal and dual are strictly feasible, then the duality gap is zero; this is the Slater condition for SDPs. Furthermore, a primal-dual point (X, y) satisfies the necessary and sufficient KKT conditions if X is primal feasible, y is dual feasible, and

$$X(C + \mathcal{A}^*(y)) = 0$$

Semidefinite programming has a rich history in the literature. See the excellent survey paper [VB94] and the references within for more details. Here, we outline a few interesting semidefinite programming formulations. For more formulations see e.g. [VB94, BPT12].

Max-cut. Given an undirected weighted graph $G = (V, E, W)$, the max-cut problem is to find a partition of the vertices such that the sum of the weights of the edges crossing the partition is maximized. This is formulated as the following optimization problem.

$$\max_{C \subseteq V} \sum_{\substack{(i,j) \in E, \\ i \in C, j \in V \setminus C}} W_{ij}$$

The max-cut problem is APX-hard [PY88]. The standard 0.879-approximation algorithm [GW94] solves the following SDP

$$\max_{X \in \mathcal{S}^n} \frac{1}{4} \langle L_G, X \rangle : X_{ii} = 1, X \succeq 0 \tag{3}$$

where $L_G = D - A$ is the weighted graph Laplacian.

Nuclear norm minimization. Nuclear norm minimization is a popular heuristic for solving inverse problems. We present a simplified form. Let $\Omega \subseteq \{1, ..., m\} \times \{1, ..., n\} \times \mathbb{R}$ denote known entries of an unknown $m \times n$ matrix W. In order to recover W, the nuclear norm heuristic solves the following problem.

$$\min_{W \in \mathbb{R}^{m,n}} \|W\|_* : W_{ij} = m_{ij}, (i, j, m_{ij}) \in \Omega$$

Here, $\|W\|_* = \sum_{i=1}^{\min(m,n)} \sigma_i(W)$. This turns out to be equivalent to the following SDP

$$\min_{\substack{X_1 \in \mathcal{S}^m, X_2 \in \mathcal{S}^n, \\ W \in \mathbb{R}^{m,n}}} \frac{1}{2} \left(\mathbf{tr}(X_1) + \mathbf{tr}(X_2) \right) : \begin{bmatrix} X_1 & W \\ W^T & X_2 \end{bmatrix} \succeq 0, W_{ij} = m_{ij}, (i, j, m_{ij}) \in \Omega$$

Furthermore, for many well conditioned inverse problems, this heuristic actually provides exact recovery with high probability (see e.g. [RFP10, CR12]).

1.2 Primal-dual interior point methods

The most robust and widely used algorithms to solve SDPs are of the class of methods typically referred to as *interior point methods*. These are the methods implemented in commercial solvers such as Mosek, SeDuMi, and CVX. Here, we present a brief overview. See e.g. [AHO98, Gu97, HRVW94] for a more thorough treatment. We first start by re-writing (2) as

$$\max_{y \in \mathbb{R}^m, Z \in \mathcal{S}^n} b^T y : \mathcal{A}^*(y) + Z = C, Z \succeq 0 \tag{4}$$

Now fix a $\mu > 0$. Given a set of points (X^μ, y^μ, Z^μ) which satisfy the feasibility constraints of (1) and (4) and also satisfies the constraint

$$X^\mu Z^\mu = \mu I$$

we say that (X^μ, y^μ, Z^μ) lies on the *central path*. It turns out (see e.g. [KSH97]) that for every μ the point (X^μ, y^μ, Z^μ) is unique, and therefore the map $\phi : \mathbb{R}_{>0} \longrightarrow \mathcal{S}^n \times \mathbb{R}^m \times \mathcal{S}^n$, $\phi(\mu) = (X^\mu, y^\mu, Z^\mu)$ is well-defined. Furthermore, if X_* is optimal for (1) and (y_*, Z_*) is optimal for (4), then

$$\lim_{\mu \downarrow 0} \phi(\mu) = (X_*, y_*, Z_*)$$

At a high level, a primal-dual interior point method simply follows the central path, generating a sequence of points $\{(X_t, y_t, Z_t)\}_{t \geq 0}$ approximately following the central path which converges to the optimal solution. The most common way this is done is to use Newton's method to find the unique solution (X_t, y_t, Z_t) to the following system of equations

$$\begin{bmatrix} \mathcal{A}^*(y_t) + Z_t - C \\ \langle A_1, X_t \rangle \\ \vdots \\ \langle A_m, X_t \rangle \\ X_t Z_t - \mu_t I \end{bmatrix} = 0$$

It can be shown that to solve this system of equations takes roughly $\mathcal{O}\left(mn^3 + m^2 n^2\right)$ operations. When n grows large (which is the regime of interest in this paper), this cost becomes prohibitive. To illustrate this point, Figure 1 shows how the runtime of Mosek grows on the max-cut SDP over random graphs on a circa 2012 machine. For $n = 10^4$, the time is prohibitive.

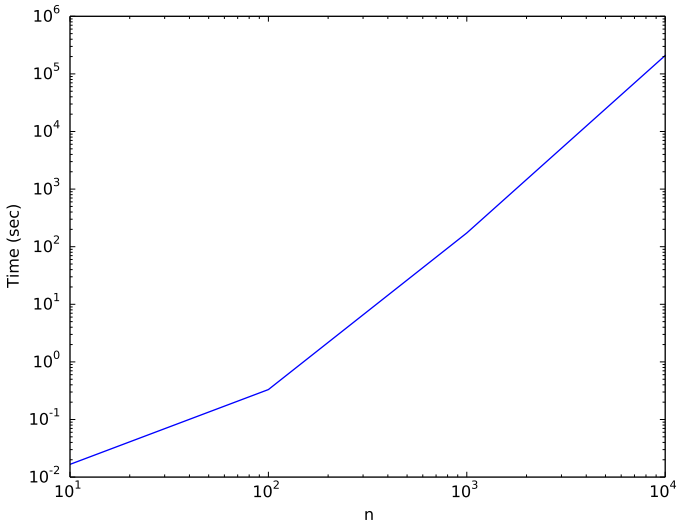

Figure 1: Plot of the time it takes Mosek to solve a random max-cut SDP instance where $|V| = n$. Note that in this experiment, roughly 30 percent of the entries in the Laplacian L_G are non-zero.

1.3 Large data regimes

In this paper, we are interested in two data regimes:

1. Storing a $\mathcal{O}\left(n^2\right)$ matrix in memory is acceptable, but performing a expensive $\mathcal{O}\left(n^3\right)$ matrix operation, such as an eigenvalue decomposition, is prohibitive.

2. Even storing a $\mathcal{O}\left(n^2\right)$ matrix in memory is prohibitive.

Regime (2) needs no motivation, with the ubiquity of the "big-data" era. Furthermore, with modern computing trends, regime (1) is also quite interesting since, for example, storing a 10000×10000

matrix in double precision takes less than 1 GB (763 MB to be precise), which is well within the limits of memory on a personal machine. However, on one of the author's modern desktop, computing a single eigen-decomposition of a symmetric 10000×10000 matrix using the LAPACK routine `syevr` takes almost 9 minutes!

Without any more assumptions, both regimes seem hopeless. Hence, we will primarily be interested in problems with exhibit the following structure

1. The matrices C and A_i, $i = 1, ..., m$, are relatively sparse.
2. The optimal solution X_* to (1) is low-rank.

As we investigate various methods, we will constantly refer back to how a method can be adapted to take advantage of both sparsity and low-rank solutions.

2 A review of first order methods

We review two common first order methods: the projected sub-gradient method, and the method of multipliers (often referred to as the augmented Lagrangian method).

2.1 Projected sub-gradient method

Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a convex function and $\mathcal{C} \subset \mathbb{R}^n$ a non-empty closed convex set. Recall that the sub-differential of f at a point $x \in \mathbb{R}^n$, denoted $\partial f(x)$, is defined as

$$\partial f(x) = \{g \in \mathbb{R}^n : f(y) \geq f(x) + \langle g, y - x \rangle \text{ for all } y \in \mathbb{R}^n\}$$

Denote by $\mathcal{P}_{\mathcal{C}}$ the Euclidean projection onto the set $\mathcal{C}$, that is

$$\mathcal{P}_{\mathcal{C}}(x) = \operatorname*{argmin}_{y \in \mathcal{C}} \|x - y\|$$

We are interested in solving general optimization problems of the form

$$\min_{x \in \mathcal{C}} f(x) \tag{5}$$

A well-known, standard algorithm for solving (5) is shown in Algorithm 1. Here, we assume that

function ProjectedSubgradientMethod(T, $\{\alpha_t\}_{t=0}^{T-1}$)
    $x_0 \leftarrow 0$
    for $t \in [0, ..., T-1]$ **do**
        $g \leftarrow$ an element of $\partial f(x_t)$.
        $x_{t+1} \leftarrow \mathcal{P}_{\mathcal{C}}(x_t - \alpha_t \frac{g}{\|g\|})$.
    return x_T.

Algorithm 1: The standard projected sub-gradient method.

(a) given an arbitrary point x we can easily compute an element of its sub-differential $\partial f(x)$, and (b) the operator $\mathcal{P}_{\mathcal{C}}$ is also easy to compute. From Nesterov, we have the following convergence result. Recall that a function f is called M-Lipschitz continuous on $\mathcal{C}$ if for every $x, y \in \mathcal{C}$ we have

$$|f(x) - f(y)| \leq M \|x - y\|$$

Theorem 1. *([Nes98], Theorem 3.2.2) Suppose f is M-Lipschitz continuous on $\mathcal{C}$ and $\|x_*\| \leq R$, where x_* is the optimal solution to (5). Then if $\alpha_t = R/\sqrt{T}$, we have*

$$f(x_T) - f(x_*) \leq \frac{MR}{\sqrt{T}}$$

Hence, to get $f(x_T) - f(x_*) \leq \epsilon$, we require $\mathcal{O}\left(1/\epsilon^2\right)$ iterations.

2.2 Method of multipliers

We now consider the following problem

$$\min_{x \in \mathbb{R}^n} f(x) : h(x) = 0 \tag{6}$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and $h : \mathbb{R}^n \longrightarrow \mathbb{R}^m$. Note that this is more general than (5), since given a set $\mathcal{C}$ we can set $h(x) = \mathbb{I}_{\mathcal{C}}(x)$, where $\mathbb{I}_{\mathcal{C}}(\cdot)$ is the indicator function on the set $\mathcal{C}$.

Fix a $\mu > 0$, and define the augmented Lagrangian $\mathcal{L}_\mu(x, \lambda)$ as

$$\mathcal{L}_\mu(x, \lambda) = f(x) + \lambda^T h(x) + \frac{\mu}{2} \|h(x)\|^2$$

The method of multipliers proceeds as shown in Algorithm 2.

function MethodOfMultipliers(T, $\{\mu_t\}_{t=0}^{T-1}$)
    Initialize (x_0, λ_0).
    for $t \in [0, ..., T-1]$ **do**
        $x_{t+1} \leftarrow \text{argmin}_{x \in \mathbb{R}^n} \mathcal{L}_{\mu_t}(x_t, \lambda_t)$.
        $\lambda_{t+1} \leftarrow \lambda_t + \mu_t h(x_t)$.
    return x_T.

Algorithm 2: The method of multipliers.

The method of multipliers is simply an algorithm which iteratively solves, by alternating between x and λ, the following saddle-point problem

$$\min_{x \in \mathbb{R}^n} \max_{\lambda \in \mathbb{R}^m} f(x) + \lambda^T h(x) + \frac{\mu}{2} \|h(x)\|^2 \tag{7}$$

It is not hard to see that an optimal solution to (7) is also an optimal solution to

$$\min_{x \in \mathbb{R}^n} \max_{y \in \mathbb{R}^m} f(x) + \lambda^T h(x) = \min_{x \in \mathbb{R}^n} f(x) : h(x) = 0$$

Unfortunately, the convergence results for the method of multipliers require quite a few assumptions, and often fail to capture rates. See e.g. Proposition 2.7 of [Ber96] for a general convergence proof. Here, we outline a standard style of convergence proof for augmented Lagrangian problems, following the style of [RFP10], Theorem 5.2.

Proposition 1. *Suppose f, h are both differentiable with continuous first derivatives, and suppose $h : \mathbb{R}^n \longrightarrow \mathbb{R}$. Let $\{(x_k, \lambda_k)\}_{k \geq 0}$ be a sequence of iterates such that*

(a) x_k *is converging to some* x_*,

(b) (x_k, λ_k) *is a stationary point of the augmented Lagrangian* $\mathcal{L}_{\mu_k}(x_k, \lambda_k)$ *for each* $k \geq 0$,

(c) $\nabla h(x_*) \neq 0$, *and*

(d) $\limsup_{k\to\infty} |\lambda_k| < \infty$.

Let $\mu_k \uparrow \infty$. *Then there exists a* $\hat{\lambda}_*$ *such that the point* $(x_*, \hat{\lambda}_*)$ *satisfies the KKT conditions for (6).*

Remark. In order to have that $\lambda_k \longrightarrow \hat{\lambda}_*$, we need additional assumptions controlling the rate of growth of $\mu_k \uparrow \infty$ versus how fast $h(x_k) \longrightarrow h(x_*)$.

Proof. Let i_* be such that $(\nabla h(x_*))_{i_*} \neq 0$. Since $\nabla h(\cdot)$ is continuous by hypothesis, and since $x_k \longrightarrow x_*$, we have $\nabla h(x_k) \longrightarrow \nabla h(x_*)$, and hence there exists a K such that for all $k \geq K$, we have $(\nabla h(x_k))_{i_*} \neq 0$. From here onwards, assume $k \geq K$. Now because each (x_k, λ_k) is stationary, we have that

$$0 = \nabla f(x_k) + \hat{\lambda}_k \nabla h(x_k) \tag{8}$$

holds for every k, where $\hat{\lambda}_k \stackrel{\text{def}}{=} \lambda_k + \mu_k h(x_k)$. Since $\nabla h(x_k) \neq 0$, $\hat{\lambda}_k$ is the unique value which satisfies (8). Then clearly $\hat{\lambda}_k = -(\nabla f(x_k))_{i_*}/(\nabla h(x_k))_{i_*}$, and by continuity we have $\hat{\lambda}_k \longrightarrow \hat{\lambda}_* \stackrel{\text{def}}{=} -(\nabla f(x_*))_{i_*}/(\nabla h(x_*))_{i_*}$, which is some finite value. But since $\mu_k \uparrow \infty$, this means that $h(x_*) = 0$ otherwise $\hat{\lambda}_*$ would be infinite. Hence, the point x_* is feasible. Once again, by continuity, taking the limit as $k \to \infty$ of both sides of (8) yields that $0 = \nabla f(x_*) + \hat{\lambda}_* \nabla h(x_*) = \nabla_x \mathcal{L}(x_*, \hat{\lambda}_*)$. This verifies the KKT conditions for $(x_*, \hat{\lambda}_*)$. □

In practice, the tricky part about this method is picking the right update method for λ_t. An alternate, more practical method than the one shown in Algorithm 2, advocated by [BM03b], is to pick a $\gamma > 1$ and $\eta < 1$ and keep an v_t around, by performing the following at the end of each iteration:

(i) Set $v \leftarrow \|h(x_t)\|^2$.

(ii) If $v < \eta v_t$ then update

$$\begin{aligned} \lambda_{t+1} &\leftarrow \lambda_t + \mu_t h(x_t) \\ \mu_{t+1} &\leftarrow \mu_t \\ v_{t+1} &\leftarrow v \end{aligned}$$

(iii) Otherwise, update

$$\begin{aligned} \lambda_{t+1} &\leftarrow \lambda_t \\ \mu_{t+1} &\leftarrow \gamma \mu_t \\ v_{t+1} &\leftarrow v_t \end{aligned}$$

3 Renegar's first-order method

3.1 Strawman

At first, it may be tempting to apply Algorithm 1 to the primal SDP (1). Certainly, the set

$$\mathcal{C} = \{X \in \mathcal{S}^n : \mathcal{A}(X) = b, X \succeq 0\}$$

is a closed, convex set. The problem, however, is that $\mathcal{P}_{\mathcal{C}}$ is an expensive operator. For instance, if $m = 1$ and $A_1 = I_n$, then we have

$$\mathcal{P}_{\mathcal{C}}(Y) = \operatorname*{argmin}_{X \in \mathcal{S}^n} \|Y - X\|_F : \mathbf{tr}(X) = 1, X \succeq 0$$

The most straight-forward way to compute $\mathcal{P}_{\mathcal{C}}$ is to take the eigen-decomposition of Y and massage the eigenvalues to live on the $(n-1)$-simplex. As we argued before, this is prohibitive. Furthermore, for a general $\mathcal{A}$, there may be no closed form solution, and computing $\mathcal{P}_{\mathcal{C}}$ then requires solving an optimization problem which is just as hard as the original problem.

3.2 Renegar's transformed problem

A trick, as observed by [Ren14], is to transform (1) into an equivalent saddle-point eigenvalue problem. For now, assume that $\mathcal{A}(I_n) = b$, that is, the identity is strictly feasible. Fix a γ such that $\gamma < \mathbf{tr}(C)$. Consider the following problem

$$\max_{X \in \mathcal{S}^n} \lambda_{\min}(X) : \mathcal{A}(X) = b, \langle C, X \rangle = \gamma \tag{9}$$

Furthermore, define the map $Z(X)$ as

$$Z(X) = I_n + \frac{1}{1 - \lambda_{\min}(X)}(X - I_n)$$

Note that $Z(X)$ is the intersection of the ray $X + tI$ with the boundary of the positive semi-definite cone.

Theorem 2. *([Ren14], Theorem 2.2) Let $\mathcal{A}(I_n) = b$ and $\gamma < \mathbf{tr}(C)$. Let X_* be optimal for (9). Then $Z(X_*)$ is optimal for (1). Hence, solving (9) solves (1).*

At a first glance, this appears to be un-interesting, since we simply traded the constraint $X \succeq 0$ with a smooth objective for a non-smooth objective with an affine constraint. We will now see why (9) is much more amenable to a projected sub-gradient method.

3.3 A projected sub-gradient algorithm

Define the sets

$$\mathcal{C} = \{X \in \mathcal{S}^n : \mathcal{A}(X) = b, \langle C, X \rangle = \gamma\}$$
$$\mathcal{C}^{\perp} = \{X \in \mathcal{S}^n : \mathcal{A}(X) = 0, \langle C, X \rangle = 0\}$$

We will now argue that $\mathcal{P}_{\mathcal{C}}$ is cheap, by arguing that $\mathcal{P}_{\mathcal{C}}^{\perp}$ is cheap: note that if $X \in \mathcal{C}$ and $V \in \mathcal{S}^n$, then $\mathcal{P}_{\mathcal{C}}(X + V) = X + \mathcal{P}_{\mathcal{C}}^{\perp}(V)$.

Proposition 2. *Consider the optimization problem*

$$\mathcal{P}_{\mathcal{C}}^{\perp}(Z) = \underset{X \in \mathcal{S}^n}{\operatorname{argmin}} \frac{1}{2} \|Z - X\|_F^2 : \mathcal{A}(X) = 0, \langle C, X \rangle = 0 \tag{10}$$

The optimal solution X_* is unique, and given by

$$X_* = Z - \mathcal{A}^*(y_*) - \lambda_* C$$

where $(y_*, \lambda_*) \in \mathbb{R}^{m+1}$ is a solution to the following $(m+1) \times (m+1)$ linear system of equations

$$\begin{bmatrix} \langle A_1, A_1 \rangle & \dots & \langle A_1, A_m \rangle & \langle A_1, C \rangle \\ & \vdots & & \vdots \\ \langle A_m, A_1 \rangle & \dots & \langle A_m, A_m \rangle & \langle A_m, C \rangle \\ \langle C, A_1 \rangle & \dots & \langle C, A_m \rangle & \langle C, C \rangle \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \\ \lambda \end{bmatrix} = \begin{bmatrix} \langle A_1, Z \rangle \\ \vdots \\ \langle A_m, Z \rangle \\ \langle C, Z \rangle \end{bmatrix}$$

Proof. It is easy to verify that $\mathcal{C}^{\perp}$ is a subspace of $\mathbb{R}^{m,m}$, and therefore its orthogonal projection $\mathcal{P}_{\mathcal{C}}^{\perp}$ is unique. Now introduce dual variables $(y, \lambda) \in \mathbb{R}^{m+1}$ and consider the Lagrangian for (10)

$$\mathcal{L}(X, y, \lambda) = \frac{1}{2} \|Z - X\|_F^2 + \langle y, \mathcal{A}(X) \rangle + \lambda \langle C, X \rangle$$

From this, we write the KKT conditions for optimal (X_*, y_*, λ_*) as

$$\begin{aligned} X_* &= Z - \mathcal{A}^*(y_*) - \lambda_* C \\ 0 &= \langle A_i, X_* \rangle, \ i = 1, ..., m \\ 0 &= \langle C, X_* \rangle \end{aligned}$$

from which the result follows immediately. □

Hence, computing $\mathcal{P}_{\mathcal{C}^{\perp}}$ is practical since we typically assume $m \ll n^2$, so we just need to solve a small system of linear equations $Ax = b$ for which A is positive semi-definite and can be pre-computed. For the max-cut problem, we can actually get a closed form solution. Define the following operator $\mathcal{Z}_0 : \mathbb{R}^{m,n} \longrightarrow \mathbb{R}^{m,n}$ which zeros out the diagonal

$$\mathcal{Z}_0(X) = X - \operatorname{diag}(\operatorname{diag}(X))$$

Corollary 1. *For the max-cut SDP, the solution to $\mathcal{P}_{\mathcal{C}^{\perp}}$ is given by*

$$\mathcal{P}_{\mathcal{C}^{\perp}}(Z) = \mathcal{Z}_0 \left\{ Z - \frac{\langle \mathcal{Z}_0(L_G), \mathcal{Z}_0(L_G) \rangle}{\langle \mathcal{Z}_0(L_G), \mathcal{Z}_0(Z) \rangle} L_G \right\}$$

To have an implementable algorithm we need one more piece, which is a method for computing a element of $\partial \lambda_1(X)$. Fortunately, this is easy by the following fact.

Proposition 3. *Let $X \in \mathcal{S}^n$, and let u be any normalized eigenvector corresponding to the eigenvalue $\lambda_1(X)$. Then $uu^T \in \partial \lambda_1(X)$.*

Proof. Recall by the variational characterization (Rayleigh-Ritz) of $\lambda_1(X)$ and the cyclic properties of the trace,

$$\lambda_1(X) = \sup_{u:\|u\|=1} u^T X u = \sup_{u:\|u\|=1} \langle uu^T, X \rangle$$

Hence, if u is a normalized eigenvector corresponding to $\lambda_1(X)$, then $\langle uu^T, X \rangle = \lambda_1(X)$. Therefore, for an arbitrary $Y \in \mathcal{S}^n$,

$$\lambda_1(Y) = \sup_{v:\|v\|=1} \langle vv^T, Y \rangle \geq \langle uu^T, Y \rangle = \langle uu^T, Y \rangle + \lambda_1(X) - \langle uu^T, X \rangle = \lambda_1(X) + \langle uu^T, Y - X \rangle$$

Hence we have verified that $uu^T \in \partial\lambda_1(X)$. □

Clearly, the dominating cost per iteration of a projected sub-gradient method for (9) is the cost of computing the top eigenvector of an $n \times n$ symmetric matrix. While computing all the eigenvectors costs $\mathcal{O}\left(n^3\right)$, computing just the top eigenvalue/eigenvector pair can be done much more efficiently with a Krylov subspace method such as Lanczos, which is implemented in the software package ARPACK. The cost of this is $\mathcal{O}\left(n^2 \log n\right)$ (see the appendix of [dEK14]).

A natural question now is, does a good approximate solution to (9) translate into a good approximate solution to (1)? This is characterized by the following proposition.

Proposition 4. *([Ren14], Proposition 2.4) Let γ satisfy $\gamma < \mathbf{tr}(C)$. Let X be feasible for (1), and let X_* be optimal for (9). Put*

$$\alpha \stackrel{\text{def}}{=} \frac{\mathbf{tr}(C) - \gamma}{\mathbf{tr}(C) - \langle C, Z(X_*) \rangle}$$

Then if $\lambda_{min}(X_*) - \lambda_{min}(X) \leq \epsilon$, we have

$$\langle C, Z(X) \rangle - \langle C, Z(X_*) \rangle \leq \frac{\epsilon}{\alpha + \epsilon} (\mathbf{tr}(C) - \langle C, Z(X_*) \rangle)$$

Proof. The proof follows immediately from observing that

$$\frac{\langle C, Z(X) \rangle - \langle C, Z(X_*) \rangle}{\mathbf{tr}(C) - \langle C, Z(X_*) \rangle} = \frac{\lambda_{\min}(X_*) - \lambda_{\min}(X)}{1 - \lambda_{\min}(X)}$$

and the fact that (see [Ren14], Lemma 2.3)

$$\lambda_{\min}(X_*) = \frac{\gamma - \langle C, Z(X_*) \rangle}{\mathbf{tr}(C) - \langle C, Z(X_*) \rangle}$$

□

3.4 Some drawbacks

Suppose we are told ahead of time that the optimal solution Z_* to the primal SDP has $\text{rank}(Z_*) = r$, where $r \ll n$. We will see later how this information can be exploited algorithmically. Unfortunately, this low rank structure is not preserved in (9).

Proposition 5. *If the optimal solution Z_* to the primal SDP has* $\operatorname{rank}(Z_*) = r$*, then the optimal solution X_* to (9) has* $\operatorname{rank}(X_*) \geq n - r$*, unless $\gamma = \langle C, Z_* \rangle$.*

Proof. From Theorem 2.2 of [Ren14], we have that if Z_* is optimal for the primal problem, then

$$X_* = I_n + \frac{\mathbf{tr}(C) - \gamma}{\mathbf{tr}(C) - \langle C, Z_* \rangle}(Z_* - I_n)$$

is optimal for (9). Hence, $X_* = (1 - \alpha)I_n + \alpha Z_*$ for some $\alpha > 0$. By assumption, $n - r$ of the eigenvalues of Z_* are zero, and since we assumed $\alpha \neq 1$, the claim follows. □

Another drawback with this approach is the case of when I_n is not strictly feasible for the primal problem. The solution proposed by Renegar is to consider a rotated system where the identity is feasible. Suppose $E \succ 0$ is feasible, and let $E^{1/2}$ denote its PSD square root. Then the equivalent problem is

$$\max_{X \in \mathcal{S}^n} \lambda_{\min}(X) : \mathcal{A}(E^{1/2} X E^{1/2}) = b, \langle E^{1/2} C E^{1/2}, X \rangle = \gamma$$

where $\gamma < \langle C, E \rangle$. There are a few potential issues that arise from this formulation. The most obvious is that computing $E^{1/2}$ requires a full eigen-decomposition (albeit we only need to do it once). Another issue is sparsity: if our original matrices A_i and C are sparse, the rotated matrices $E^{1/2} A_i E^{1/2}$ and $E^{1/2} C E^{1/2}$ won't necessarily preserve sparsity.

4 Dual ascent methods

Section 3 explored optimizing the primal problem. Recalling that the Slater condition ensures us strong duality for strictly feasible SDPs, we now explore a class of methods which operate on the dual formulation. This is particular attractive for large data regimes, since the dual decision variable lives in $\mathbb{R}^m$.

4.1 Motivation and setup

Recall the dual problem is

$$\max_{y \in \mathbb{R}^m} -b^T y : C + \mathcal{A}^*(y) \succeq 0$$

This is equivalent to

$$\max_{y \in \mathbb{R}^m} -b^T y : \lambda_n(C + \mathcal{A}^*(y)) \geq 0$$

Since the objective function is affine in y, the maximum is attained on the boundary of the feasible region. That is, the problem is equivalent to

$$\max_{y \in \mathbb{R}^m} -b^T y : \lambda_n(C + \mathcal{A}^*(y)) = 0 \iff \max_{y \in \mathbb{R}^m} b^T y : \lambda_1(\mathcal{A}^*(y) - C) = 0 \tag{11}$$

4.2 A method of multipliers approach

At this point, it may be tempting to apply the method of multipliers to (11). The augmented Lagrangian would be

$$\mathcal{L}_\mu(y,\nu) = b^T y + \nu\lambda_1(\mathcal{A}^*(y) - C) + \frac{\sigma}{2}\lambda_1(\mathcal{A}^*(y) - C)^2$$

The issue here is that $y \mapsto \frac{\sigma}{2}\lambda_1(\mathcal{A}^*(y) - C)^2$ is neither differentiable nor convex (so the notion of a sub-gradient does not even make sense). Hence, $\operatorname{argmax}_y \mathcal{L}_\mu(y,\nu)$ would be tough to deal with numerically. One way to deal with this issue is to use a smooth approximation to $\lambda_1(X)$, introduced by Nesterov.

Lemma 1. *([Nes04], Section 4) Let $X \in \mathcal{S}^n$, and fix a $\mu > 0$. The function $f_\mu(X)$ given by*

$$f_\mu(X) = \mu \log\left\{\sum_{i=1}^n \exp(\lambda_i(X)/\mu)\right\}$$

is convex and twice differentiable with gradient

$$\nabla f_\mu(X) = \left(\sum_{i=1}^n \exp(\lambda_i(X)/\mu)\right)^{-1} \sum_{i=1}^n \exp(\lambda_i(X)/\mu) q_i q_i^T$$

where q_i is the i-th column of the unitary matrix Q in the eigen-decomposition $Q\Sigma Q^T$ of X. Furthermore, $f_\mu(X)$ satisfies the inequalities

$$\lambda_1(X) \le f_\mu(X) \le \lambda_1(X) + \mu \log n$$

Hence $\lim_{\mu \downarrow 0} f_\mu(X) = \lambda_1(X)$.

Therefore, we apply the method of multipliers to the smoothed variant

$$\max_{y \in \mathbb{R}^m} b^T y : f_\mu(\mathcal{A}^*(y) - C) = 0 \tag{12}$$

From this lemma, for μ arbitrarily close to 0, we have the following approximations

$$f_\mu(X) \approx \lambda_1(X)$$
$$\nabla f_\mu(X) \approx \frac{1}{k}\sum_{i=1}^k q_i q_i^T$$

where k is the multiplicity of the top eigenvalue.

4.3 A precursor to spectral bundle methods

We now focus on a particular subclass of SDPs where we can turn the dual problem into an *unconstrained* minimization problem. This approach is directly inspired by the spectral bundle work (see e.g. [HR00]) Here, we make the following assumption: suppose that whenever $\mathcal{A}(X) = b$ holds, then $\langle E, X\rangle = \alpha$ holds, where $E \succ 0$ and α is some constant. An example of this is the max-cut SDP, where $E = I_n$ and $\alpha = n$. Under this assumption, we can derive the following equivalent unconstrained dual.

Lemma 2. *Suppose that $\mathcal{A}(X) = b$ implies that $\langle E, X\rangle = \alpha$ for some $E \succ 0$. Then, solving the following unconstrained problem is equivalent to solving the dual SDP (2)*

$$\max_{y\in\mathbb{R}^m} -y^T b + \alpha\lambda_n(\widetilde{C} + \widetilde{\mathcal{A}}^*(y))$$

where $\widetilde{C} = E^{-1/2}CE^{-1/2}$, $\widetilde{A}_i = E^{-1/2}A_iE^{-1/2}$ for $i = 1, ..., m$, and

$$\widetilde{\mathcal{A}}(X) = \begin{bmatrix} \langle \widetilde{A}_1, X\rangle \\ \vdots \\ \langle \widetilde{A}_m, X\rangle \end{bmatrix}$$

Proof. We write the primal SDP, adding in the additional redundant constraint, which does not change the problem

$$\min_{X\in\mathcal{S}^n} \langle C, X\rangle : \mathcal{A}(X) = b, \langle E, X\rangle = \alpha, X \succeq 0$$

Noting that $\langle E, X\rangle = \langle I_n, E^{1/2}XE^{1/2}\rangle$ via the cyclic properties of trace, we change variables via the transformation $\widetilde{X} = E^{1/2}XE^{1/2}$. Hence, we have the equivalent problem

$$\min_{\widetilde{X}\in\mathcal{S}^n} \langle C, E^{-1/2}\widetilde{X}E^{-1/2}\rangle : \mathcal{A}(E^{-1/2}\widetilde{X}E^{-1/2}) = b, \langle I_n, \widetilde{X}\rangle = \alpha, E^{-1/2}\widetilde{X}E^{-1/2} \succeq 0$$

Noting that $E^{-1/2}\widetilde{X}E^{-1/2} \succeq 0 \iff \widetilde{X} \succeq 0$, this yields the equivalent problem

$$\min_{X\in\mathcal{S}^n} \langle \widetilde{C}, X\rangle : \widetilde{\mathcal{A}}(X) = b, \langle I_n, X\rangle = \alpha, X \succeq 0$$

Introducing multipliers (y, S, ν), the Lagrangian for this problem is

$$\mathcal{L}(X, y, S, \nu) = \langle \widetilde{C}, X\rangle + \langle y, \widetilde{\mathcal{A}}(X) - b\rangle - \langle S, X\rangle + \nu(\langle I_n, X\rangle - \alpha)$$

which admits a dual of the form

$$\max_{y\in\mathbb{R}^m,\nu\in\mathbb{R}} -y^T b - \nu\alpha : \lambda_n(\widetilde{C} + \widetilde{\mathcal{A}}^*(y) + \nu I_n) = 0$$

Since $\lambda_n(\widetilde{C} + \widetilde{A}^*(y) + \nu I_n) = \lambda_n(\widetilde{C} + \widetilde{\mathcal{A}}^*(y)) + \nu$, we can eliminate ν from the dual problem and get

$$\max_{y\in\mathbb{R}^m} -y^T b + \alpha\lambda_n(\widetilde{C} + \widetilde{\mathcal{A}}^*(y))$$

which yields the result. □

An immediate corollary is that when $E = I_n$ (such as in the max-cut formulation), the dual problem has a very nice form

$$\max_{y\in\mathbb{R}^m} -y^T b + \alpha\lambda_n(C + \mathcal{A}^*(y)) \iff \min_{y\in\mathbb{R}^m} -y^T b + \alpha\lambda_1(\mathcal{A}^*(y) - C) \tag{13}$$

Notice that this form is immediately amenable to a sub-gradient method. Furthermore, since the objective is a convex function, the convergence rate of Theorem 1 applies. Additionally, the function $\lambda_1(\cdot)$ can be smoothed with the smooth function of Lemma 1 and an accelerated gradient method such as Nesterov's method used. We discuss this in more detail in Section 8. To calculate the sub-gradient of the objective function in (13), we have the following fact.

Proposition 6. *Define $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ as $f(y) = \lambda_1(\mathcal{A}^*(y) - C)$. Let u be a normalized eigenvector of $A^*(y) - C$ corresponding to $\lambda_1(A^*(y) - C)$. Then $\mathcal{A}(uu^T) \in \partial f(y)$.*

Proof. The proof is similar to Proposition 3. Observe that

$$\begin{aligned}
f(z) &= \lambda_1(\mathcal{A}^*(z) - C) \\
&= \sup_{v:\|v\|=1} \langle vv^T, \mathcal{A}^*(z) - C\rangle \\
&\geq \langle uu^T, \mathcal{A}^*(z) - C\rangle \\
&= \langle uu^T, \mathcal{A}^*(z) - C\rangle + \lambda_1(\mathcal{A}^*(y) - C) - \langle uu^T, \mathcal{A}^*(y) - C\rangle \\
&= \lambda_1(\mathcal{A}^*(y) - C) + \langle uu^T, \mathcal{A}^*(z - y)\rangle \\
&\overset{(a)}{=} f(y) + \langle \mathcal{A}(uu^T), z - y\rangle
\end{aligned}$$

where in (a) we used the fact that the adjoint operator satisfies $\langle \mathcal{A}(X), y\rangle = \langle X, \mathcal{A}^*(y)\rangle$ for all $X \in \mathcal{S}^n, y \in \mathbb{R}^m$. □

5 Explicit low-rank factorization methods

We now turn our attention back to the primal problem, but this time we are interested in exploiting the low rank structure of our SDP solution. The ideas presented in this section come mostly from [BM03b].

5.1 Formulation

Suppose we know our optimal solution has rank at most k. Then it makes sense to restrict our decision variables to matrices of rank at most k. We do this by noting the following equivalence

$$\{X \in \mathcal{S}^n : X \succeq 0, \text{rank}(X) \leq r\} = \{LL^T : L \in \mathbb{R}^{n,r}\}$$

Hence, we rewrite the primal SDP (1) as

$$\min_{L \in \mathbb{R}^{n,r}} \langle C, LL^T\rangle : \mathcal{A}(LL^T) = b$$

This form is appealing since the PSD cone constraint $X \succeq 0$ is now gone. However, we have replaced it with a non-linear constraint $\mathcal{A}(LL^T) = b$. Burer and Montiero suggest to use the method of multipliers to solve this problem, with the augmented Lagrangian

$$\mathcal{L}_\mu(L, y) = \langle C, LL^T\rangle + \langle y, \mathcal{A}(LL^T) - b\rangle + \frac{\mu}{2}\left\|\mathcal{A}(LL^T) - b\right\|^2$$

To illustrate the inherent non-convexity of this method, in Figure 2 we plot the non-convex surface for the functional $L \mapsto \mathcal{L}_\mu(L, y)$ in the simple case of $L \in \mathbb{R}^{2,1}$.

5.2 Picking the right rank

A natural question from the formulation above is, how do we pick the right value of r ahead of time? It turns out, we can upper bound the rank of a solution just based on the number of constraints we have.

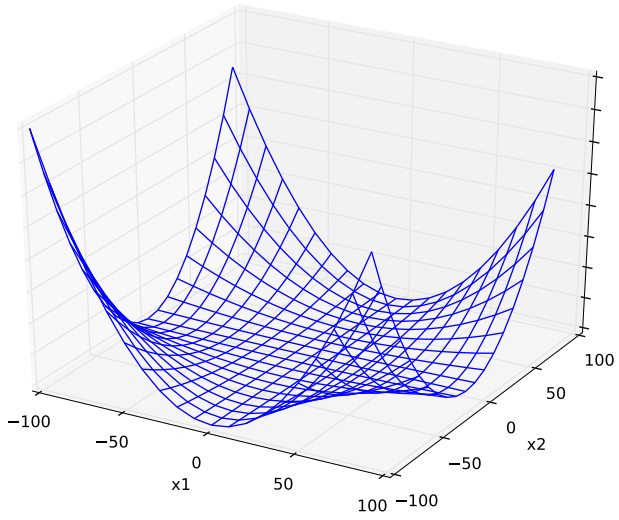

Figure 2: An example plot of the non-convex surface of $L \mapsto \mathcal{L}_\mu(L, y)$ for the case when $L \in \mathbb{R}^{2,1}$.

Theorem 3. *([MNP98], Theorem 1.1) Let X be an extreme point of the set $\{X \in \mathcal{S}^n : \mathcal{A}(X) = b, X \succeq 0\}$. Put $r = \operatorname{rank}(X)$. Then $r(r+1)/2 \leq m$.*

Proof. Let $X = Q\Sigma Q^T$ denote the eigen-decomposition of X, with $Q \in \mathbb{R}^{n,r}$ and $\Sigma \in \mathcal{S}^n$. Since X is feasible, we have for $i = 1, ..., m$ that

$$\langle A_i, X \rangle = \langle Q^T A_i Q, \Sigma \rangle = b_i$$

Define the linear operator $\mathcal{B} : \mathcal{S}^r \longrightarrow \mathbb{R}^m$ as

$$\mathcal{B}(X) = \begin{bmatrix} \langle Q^T A_1 Q, X \rangle \\ \vdots \\ \langle Q^T A_m Q, X \rangle \end{bmatrix}$$

Now assume for a contradiction that $r(r+1)/2 > m$. By the rank-nullity theorem, we have that

$$\begin{aligned} \dim(\operatorname{kern} \mathcal{B}) &= \dim(\mathcal{S}^r) - \dim(\operatorname{im} \mathcal{B}) \\ &= r(r+1)/2 - \dim(\operatorname{im} \mathcal{B}) \\ &\geq r(r+1)/2 - m \\ &> 0 \end{aligned}$$

Hence, the null space of $\mathcal{B}$ is not trivial, so there exists a $V \in \mathcal{S}^r$, $V \neq 0$ such that $\mathcal{B}(V) = 0$. Note we can find ϵ small enough such that both $\Sigma + \epsilon V \succeq 0$ and $\Sigma - \epsilon V \succeq 0$. An easy sufficient condition is to set $\epsilon \leq \Sigma_r / \min(|\lambda_1(V)|, |\lambda_r(V)|)$. Pick such an ϵ, and set $X_1 = Q(\Sigma + \epsilon V)Q^T$ and $X_2 = Q(\Sigma - \epsilon V)Q^T$. It is easy to verify that

(i) $X_1 \neq X_2$.

(ii) $X = \frac{1}{2}(X_1 + X_2)$.

(iii) $\mathcal{A}(X_1) = \mathcal{A}(X_2) = b$.

(iv) $X_1 \succeq 0$, $X_2 \succeq 0$.

But this directly contradicts the fact that X is an extreme point. □

From this, we have immediately the following result.

Corollary 2. *There exists an optimal X_* for (1) such that $r = \operatorname{rank}(X_*)$ satisfies $r(r+1)/2 \leq m$.*

Proof. This follows immediately from the fact that for SDP, at least one optimal solution will be an extreme point. □

Hence, we can always safely set $r = \mathcal{O}(\sqrt{m})$. We can, however, do slightly better. We can start with a small r, and repeat the following while $r \leq \bar{r}$, where $\bar{r}$ is the smallest integer satisfying $\bar{r}(\bar{r}+1)/2 \geq m$.

(1) Compute (R_r, y_r) using method of multipliers.

(2) Check if $(R_r R_r^T, y_r)$ satisfy the KKT conditions. If so, an optimal point is found.

(3) Otherwise, increase r.

5.3 Convergence results

As is typical with method of multiplier algorithms, establishing a finite time convergence rate is difficult. Burer and Montiero are able to establish the following result. Let $H \mapsto \nabla^2_{LL}\mathcal{L}_\mu(L, y)[H, H]$ denote the quadratic form associated with the Hessian of $\mathcal{L}$ w.r.t. L.

Theorem 4. *([BM03a], Theorem 4.1) Suppose that the optimal X_* has rank r, and let $\{L^k\}_{k\geq 1}$ be a bounded sequence of $\mathbb{R}^{n\times r}$. Suppose the following conditions are all satisfied*

(a) $\lim_{k\to\infty} \mathcal{A}(L^k(L^k)^T) = b$.

(b) $\lim_{k\to\infty} \nabla_L \mathcal{L}_{\mu_k}(L^k, y^k) = 0$.

(c) $\liminf_{k\to\infty} \nabla^2_{LL}\mathcal{L}_{\mu_k}(L^k, y^k)[H^k, H^k] \geq 0$ *for all bounded sequences* $\{H^k\}_{k\geq 1}$ *of* $\mathbb{R}^{n\times r}$.

(d) $\operatorname{rank}(R^k) < r$ *for* $k \geq 1$.

Then every limit point of $\{L^k(L^k)^T\}_{k\geq 1}$ is optimal for (1).

While this result holds for any SDP, it is not immediately obvious how it can be applied. For instance, condition (c) seems difficult to check in practice, and more discomforting is that we only ever run for finite time. Unfortunately, deriving a general theory for non-convex optimization is elusive; instead we have to focus on specific problem instances. Recently, the empirical success of these low rank factorization methods has sparked research interest, and there have been many works (see e.g. [JNS13, NJS13, NNS+14, FXY13]) which establish finite time convergence results for a low rank factorization method on specific problem instances of interest (which can be formulated as semi-definite programs).

Matrix sensing via low rank factorization. We illustrate the low rank factorization method applied to a problem similar to matrix completion. Suppose there exists an unknown matrix $M \in \mathbb{R}^{m,n}$, from which we get to observe p measurements $b_i = \langle A_i, M \rangle$, $i = 1, ..., p$ for some known collection of $\{A_i\}_{i=1}^p$. We want to solve the following feasibility problem

$$\min_{X \in \mathbb{R}^{m,n}} 0 : \mathcal{A}(X) = b, \operatorname{rank}(X) \leq k$$

where $\mathcal{A}(X)_i = \langle A_i, X \rangle$ and $b = (b_1, ..., b_p)^T$. Note that matrix completion is a special case of this problem, with $A_i = e_{a_i} e_{b_i}^T$ for index sets $\{a_i\}$ and $\{b_i\}$ (ignoring the nuclear norm penalty). We can solve this problem iteratively by factorizing $X = UV^T$, where $U \in \mathbb{R}^{m,k}$ and $V \in \mathbb{R}^{n,k}$, and iteratively computing

$$V^{t+1} \leftarrow \operatorname*{argmin}_{V \in \mathbb{R}^{n,k}} \left\| \mathcal{A}(U^t (V^t)^T) - b \right\|^2$$
$$U^{t+1} \leftarrow \operatorname*{argmin}_{U \in \mathbb{R}^{m,k}} \left\| \mathcal{A}(U^t (V^{t+1})^T) - b \right\|^2$$

It turns out, it is possible to actually prove convergence results for this non-convex algorithm, for random instances of this problem. One particular result is as follows.

Theorem 5. *([JNS13], Theorem 2.2) Let each entry in A_i be distributed i.i.d. from $\mathcal{N}(0, 1)$. Then if $p \geq \Omega(\frac{kn}{\delta_k^2} \log n)$ where $\delta_k < \left(\frac{\sigma_k(M)}{\sigma_1(M)}\right)^2 \frac{1}{100k}$, we have w.h.p. for all $t > 2 \log(\|M\|_F / \epsilon)$,*

$$\left\| M - U^t (V^t)^T \right\|_F \leq \epsilon$$

This result holds more generally for linear operators $\mathcal{A}$ which satisfy a *restricted isometry property* (RIP). See [JNS13] for more details.

6 Block coordinate descent methods

In this section, we focus on a class of methods which applies the idea of block coordinate descent to the SDP problem. The idea is quite simple: work on a particular row/column of the matrix at a time, holding all other entries constant. The ideas presented in this section are derived from [WGMS09].

6.1 Formulation

The key enabler of this technique is the following generalized Schur complement fact.

Lemma 3. *([Zha05], Theorem 1.20) Let $X \in \mathcal{S}^n$ be partitioned as*

$$X = \begin{bmatrix} a & b^T \\ b & C \end{bmatrix}$$

where $C \in \mathcal{S}^{n-1}$. Then $X \succeq 0$ iff $C \succeq 0$, $a - b^T C^\dagger b \geq 0$, and $b \in \operatorname{im} C$, where $C^\dagger$ is the Moore-Penrose pseudo-inverse of C.

This lemma is used as follows. Start with an $X \succeq 0$, which ensures us that $C \succeq 0$. Now, we are free to move a, b as we choose, as long as we keep (a) $a - b^T C^\dagger b \geq 0$ and (b) $b \in \operatorname{im} C$. To fully introduce the method, we need a bit of notation. If $A \in \mathbb{R}^{n,n}$ is a matrix and $\alpha, \beta \subseteq \{1, ..., n\}$ are index sets, then define $A_{\alpha,\beta}$ to be

$$A_{\alpha,\beta} = \begin{bmatrix} A_{\alpha_1,\beta_1} & \dots & A_{\alpha_1,\beta_{|\beta|}} \\ & \vdots & \\ A_{\alpha_{|\alpha|},\beta_1} & \dots & A_{\alpha_{|\alpha|},\beta_{|\beta|}} \end{bmatrix}$$

If $i \in \{1, ..., n\}$, define $i^c = \{1, ..., n\} \setminus \{i\}$. Finally, relabel $A^{(i)} = A_i$. Using the Schur complement lemma above and starting from a feasible X, we we can update row/column i, for $i = 1, ..., n$, by solving the following problem

$$\min_{x \in \mathbb{R}, y, z \in \mathbb{R}^{n-1}} \widetilde{c}_i^T \begin{bmatrix} x \\ y \end{bmatrix} : \widetilde{A}_i \begin{bmatrix} x \\ y \end{bmatrix} = \widetilde{b}_i,\ x - y^T X_{i^c,i^c}^\dagger y \geq 0,\ y = X_{i^c,i^c} z \tag{14}$$

where

$$\widetilde{c}_i = \begin{bmatrix} C_{ii} \\ 2C_{i^c,\{i\}} \end{bmatrix},\ \widetilde{A}_i = \begin{bmatrix} A_{ii}^{(1)} & 2A_{\{i\},i^c}^{(1)} \\ \vdots & \vdots \\ A_{ii}^{(m)} & 2A_{\{i\},i^c}^{(m)} \end{bmatrix},\ \widetilde{b}_i = \begin{bmatrix} b_1 - \langle A_{i^c,i^c}^{(1)}, X_{i^c,i^c} \rangle \\ \vdots \\ b_m - \langle A_{i^c,i^c}^{(m)}, X_{i^c,i^c} \rangle \end{bmatrix}$$

Of course, as written, this is not yet practical since (a) computing $X_{i^c,i^c}^\dagger$ requires a full eigendecomposition and (b) it is not immediately clear that the problem (14) is any easier to solve than the SDP. One immediate trick to making (14) easier to solve is to apply the augmented Lagrangian method to the linear constraint. That is, introduce a dual multiplier $\mu \in \mathbb{R}^m$ and solve the sub-problem

$$\min_{x \in \mathbb{R}, y, z \in \mathbb{R}^{n-1}} \widetilde{c}_i^T \begin{bmatrix} x \\ y \end{bmatrix} + \langle \mu, \widetilde{A}_i \begin{bmatrix} x \\ y \end{bmatrix} - \widetilde{b}_i \rangle + \frac{\sigma}{2} \left\| \widetilde{A}_i \begin{bmatrix} x \\ y \end{bmatrix} - \widetilde{b}_i \right\|^2 : x - y^T X_{i^c,i^c}^\dagger y \geq 0,\ y = X_{i^c,i^c} z$$

By defining $\widetilde{b}_i' = \widetilde{b}_i - \frac{1}{\sigma}\mu$, some algebra yields that the following problem has the same optimal solution set.

$$\min_{x \in \mathbb{R}, y, z \in \mathbb{R}^{n-1}} \widetilde{c}_i^T \begin{bmatrix} x \\ y \end{bmatrix} + \frac{\sigma}{2} \left\| \widetilde{A}_i \begin{bmatrix} x \\ y \end{bmatrix} - \widetilde{b}_i' \right\|^2 : x - y^T X_{i^c,i^c}^\dagger y \geq 0,\ y = X_{i^c,i^c} z \tag{15}$$

As we shall see now, by specializing the method to certain problems, it becomes very easy to solve either (14) or (15).

6.2 Max-cut specialization

Recall for the max-cut SDP, we have $A_i = e_i e_i^T$, where $e_i \in \mathbb{R}^n$ is the i-th standard basis vector. It is not hard to see, since the diagonal entries are constrained to be 1, that (14) reduces to

$$\min_{y \in \operatorname{im} X_{i^c,i^c}} \widehat{c}^T y : 1 - y^T X_{i^c,i^c}^\dagger y \geq 0 \tag{16}$$

where $\widehat{c} = 2C_{i^c,i}$. It turns out, (16) has a very simple closed form solution.

Lemma 4. *([WGMS09], Lemma 3.1) Fix an $i \in \{1, ..., n\}$ and put $D = X_{i^c,i^c}$. If $\widehat{c}^T D \widehat{c} > 0$, then the solution to (16) is given by $y_* = -\frac{1}{\sqrt{\widehat{c}^T D \widehat{c}}} D\widehat{c}$. Otherwise, $y_* = 0$.*

Proof. Let $\text{rank}(D) = r \leq n$ and let D have eigen-decomposition $D = Q_r \Sigma_r Q_r^T$, where $Q_r \in \mathbb{R}^{n,r}$ and $\Sigma_r \succ 0$. We re-write (16) to

$$\begin{aligned}
\min_{z \in \mathbb{R}^{n-1}} (D\widehat{c})^T z : 1 - z^T D D^\dagger D z \geq 0 &\iff \min_{z \in \mathbb{R}^{n-1}} (D\widehat{c})^T z : 1 - z^T D z \geq 0 \\
&\iff \min_{z \in \mathbb{R}^{n-1}} (Q_r^T \widehat{c})^T \Sigma_r Q_r^T z : 1 - (\Sigma_r Q_r^T z)^T \Sigma_r^{-1} (\Sigma_r Q_r^T z)^T \geq 0 \\
&\iff \min_{z_r \in \mathbb{R}^r} (Q_r^T \widehat{c})^T z_r : 1 - z_r^T \Sigma_r^{-1} z_r \geq 0 \qquad (17)
\end{aligned}$$

where the third equivalence is because Q_r is full column rank. Now we can solve (17) by simply looking at the KKT conditions. Introducing multiplier $\lambda \geq 0$, the Lagrangian for (17) is

$$\mathcal{L}(z_r, \lambda) = (Q_r^T \widehat{c})^T z_r + \lambda (z_r^T \Sigma_r^{-1} z_r - 1)$$

The stationarity condition yields that for optimal (z_r^*, λ^*),

$$0 = \nabla_{z_r}(z_r^*, \lambda^*) = Q_r^T \widehat{c} + 2\lambda^* \Sigma_r^{-1} z_r^* \Longrightarrow z_r^* = -\frac{1}{2\lambda^*} \Sigma_r Q_r^T \widehat{c} \text{ if } \lambda^* \neq 0$$

and the slackness condition yields that

$$\lambda^* ((z_r^*)^T \Sigma_r^{-1} z_r^* - 1) = 0$$

There are two cases. If $\lambda^* = 0$, then by stationarity we have $Q_r^T \widehat{c} = 0$. Otherwise, by combining the stationarity and slackness condition, we have that

$$\frac{1}{4(\lambda^*)^2} \widehat{c}^T Q_r \Sigma_r \Sigma_r^{-1} \Sigma_r Q_r^T \widehat{c} - 1 = 0 \Longrightarrow \lambda^* = \frac{1}{2} \sqrt{\widehat{c}^T D \widehat{c}}$$

and hence

$$z_r^* = -\frac{1}{\sqrt{\widehat{c}^T D \widehat{c}}} \Sigma_r Q_r^T \widehat{c}$$

To finish the proof, note we have the following variable transforms

$$z_r = \Sigma_r Q_r^T z, y = Q_r \Sigma_r Q_r^T z \Longrightarrow Q_r Z_r = y$$

and hence when $\lambda^* \neq 0$ we have $y^* = -\frac{1}{\sqrt{\widehat{c}^T D \widehat{c}}} D\widehat{c}$. Furthermore, noting that $\lambda^* = 0 \iff \widehat{c}^T D \widehat{c} = 0$ finishes the proof. □

6.3 Matrix completion specialization

For matrix completion, $C = \frac{1}{2} I_{m+n}$ and the p-th entry in Ω is associated to

$$A_p = \begin{bmatrix} 0 & e_i e_j^T \\ e_j e_i^T & 0 \end{bmatrix}$$

where $A_p \in \mathcal{S}^{m+n}$. Therefore, it is straightforward to see that the problem (15) is equivalent to

$$\min_{x\in\mathbb{R},y,z\in\mathbb{R}^{m+n-1}} \frac{1}{2}x + \frac{\sigma}{2}\left\|\widetilde{K}_i y - \widetilde{b}'_i\right\|^2 : x - y^T X^{\dagger}_{i^c,i^c} y \geq 0, y = X_{i^c,i^c} z \tag{18}$$

$$\iff \min_{x\in\mathbb{R},z\in\mathbb{R}^{m+n-1}} \frac{1}{2}x + \frac{\sigma}{2}\left\|\widetilde{K}_i X_{i^c,i^c} z - \widetilde{b}'_i\right\|^2 : x - z^T X_{i^c,i^c} z \geq 0 \tag{19}$$

where $\widetilde{K}_i$ is given as

$$\widetilde{K}_i = 2\begin{bmatrix} A^{(1)}_{\{i\},i^c} \\ \vdots \\ A^{(|\Omega|)}_{\{i\},i^c} \end{bmatrix}$$

Introducing a multiplier λ and forming the Lagrangian for (19) we get

$$\mathcal{L}(x,z,\lambda) = \frac{1}{2}x + \frac{\sigma}{2}\left\|\widetilde{K}_i X_{i^c,i^c} z - \widetilde{b}'_i\right\|^2 + \lambda(z^T X_{i^c,i^c} z - x)$$

Since $\nabla_x \mathcal{L}(x,z,\lambda) = 0 \Longrightarrow \lambda_* = 1/2$, by complementary slackness we have that $z_*^T X_{i^c,i^c} z_* - x_* = 0$, so we can solve the equivalent unconstrained problem for z_*,

$$\min_{z\in\mathbb{R}^{m+n-1}} \frac{1}{2} z^T X_{i^c,i^c} z + \frac{\sigma}{2}\left\|\widetilde{K}_i X_{i^c,i^c} z - \widetilde{b}'_i\right\|^2$$

This problem has an optimal solution $X_{i^c,i^c} z_* = y_*$ given by

$$(I_{m+n-1} + \sigma X_{i^c,i^c} \widetilde{K}_i^T \widetilde{K}_i) y_* = \widetilde{K}_i^T X_{i^c,i^c} \widetilde{b}'_i$$

Since the matrix $(I_{m+n-1} + \sigma X_{i^c,i^c} \widetilde{K}_i^T \widetilde{K}_i) \succ 0$, there is a unique y_* solution. However, as written, this form is not appealing since it amounts to solving an $(m+n-1)\times(m+n-1)$ system of linear equations. Fortunately, it turns out that this problem has a lot of structure which makes it quite tractable.

Lemma 5. *([WGMS09], Lemma 4.1) Fix an $i \in \{1,2,...,m+n\}$. Define the following sets*

$$\alpha \overset{\text{def}}{=} \begin{cases} \{j+m : (i,j) \in \Omega\} & \text{if } 1 \leq i \leq m \\ \{j-m : (j,i) \in \Omega\} & \text{if } m < i \leq m+n \end{cases} \qquad \beta \overset{\text{def}}{=} i^c \setminus \alpha$$

Suppose $|\alpha| > 0$. The optimal (x_*, y_*) for (18) is given by

$$\begin{aligned} (y_*)_\alpha &= \left(\frac{2}{\sigma} I_{|\alpha|} + X_{\alpha,\alpha}\right)^{-1} X_{\alpha,\alpha}(\widetilde{b}'_i)_\alpha \\ (y_*)_\beta &= \frac{\sigma}{2} X_{\beta,\alpha}((\widetilde{b}'_i)_\alpha - (y_*)_\alpha) \\ x_* &= \frac{\sigma}{2}(y_*)_\alpha^T((\widetilde{b}'_i)_\alpha - (y_*)_\alpha) \end{aligned}$$

where we abuse notation and let $(y_*)_\alpha$ (resp. $(y_*)_\beta$) denote the positions in y_* which correspond to the indices in α (resp. β). That is, the update rule for the i-th column of X^t is given as

$$X^{t+1}_{\alpha,i} \leftarrow (y_*)_\alpha, \qquad X^{t+1}_{\beta,i} \leftarrow (y_*)_\beta, \qquad X^{t+1}_{i,i} = x_*$$

Otherwise, if $|\alpha| = 0$, set $(x_*, y_*) = (0,0)$.

Proof. The proof is straightforward but tedious, so we omit the details. The intuition is, however, that α contains the indices which are coupled due to the measurements, for which we need to solve a linear system of equations. These indices are determined by the value of $\widetilde{K}_i^T \widetilde{K}_i$. Once we have determined the values for the indices in α, the remaining values can be easily read off. □

Note that, in the worst case, $|\alpha| = \max\{m, n\}$, which happens when either a row or column is fully measured. For a dataset, the worst case value for $|\alpha|$ is simply the maximum number of times a particular row or column is measured; for matrix completion style problems, we expect this to be much smaller than $\max\{m, n\}$. Hence, we expect solving for $(y_*)_\alpha$ to be relative cheap. Note that in doing so, forming the matrix inverse is not necessary, since the matrix $(\frac{2}{\sigma} I_{|\alpha|} + X_{\alpha,\alpha}) \succ 0$, and hence we can use a Cholesky decomposition to solve the system.

7 Evaluation

In this section we present an experimental evaluation of several of the methods presented in the previous sections.

Hardware. Our experiments are run on a machine with 80 Intel Xeon E7-8870 CPUs in an 8x10 configuration. These chips have private L1/L2 caches and a shared L3 cache of 30MB, and support SSE4 vectorized instructions. The machine has 256GB of DRAM.

Software. The core parts of each algorithm (e.g. gradient computations, matrix operations) are implemented in C++ using the high-performance `eigen3`[1] C++ matrix library. We then wrap these low level kernels using `Cython`[2] to provide a high level Python API. For top eigenvalue/eigenvector pair computations, we use the ARPACK[3] library, which provides a high performance implementation of the Lanczos algorithm. For optimization, we use the implementation of L-BFGS provided by [BLNZ95]. Hence, we believe our implementations are within a factor or two of a pure C implementation relying on the standard numerical libraries. We plan to release our code soon on github, along with a general purpose library for solving semidefinite programs with first order methods.

7.1 Renegar's method

We made a best effort to try and tune Renegar's method from Section 3 to work well on the max-cut SDP. Unfortunately, we were not able to succeed. Figure 3 shows the typical performance of Renegar's method on the max-cut SDP for a random graph instance with $|V| = 50$, which is a trivial sized graph for which an interior point solver can optimize in under a second. OPT is computed by having Mosek converge to a high degree of accuracy.

The list of tuning methods we attempted which did not yield any noticeable difference in performance are

(i) Different step sizes $\{\alpha/\sqrt{t}\}_{t\geq 1}$, $\{\alpha/t\}_{t\geq 1}$ and $\{\alpha/\sqrt{T}\}_{t\geq 1}^{T}$ for various values of $\alpha > 0$.

(ii) Using random cuts which have strictly higher objective value than $X = I$ for the initialization point.

[1] http://eigen.tuxfamily.org/index.php?title=Main_Page
[2] http://cython.org/
[3] http://www.caam.rice.edu/software/ARPACK/

function ProjectedSubgradientMethodAvg(T, $\{\alpha_t\}_{t=0}^{T-1}$)
    $x_0 \leftarrow 0$
    $y_0 \leftarrow 0$
    for $t \in [0, ..., T-1]$ **do**
        $g \leftarrow$ an element of $\partial f(x_t)$.
        $x_{t+1} \leftarrow \mathcal{P}_{\mathcal{C}}(x_t - \frac{g}{\|g\|})$.
        $y_{t+1} \leftarrow \frac{\alpha_t}{\sum_{k=0}^{t} \alpha_k} x_{t+1}$.
    return y_T.

Algorithm 3: The projected sub-gradient method with iterate averaging.

(iii) Renegar's Non-Smoothed Subscheme (see Section 4 of [Ren14]).

(iv) Iterate averaging (see Algorithm 3 and explanation below).

Algorithm 3 shows the iterate averaging scheme (which we denote **First-order-avg** in Figure 3). It is a simple modification of the projected sub-gradient method shown in Algorithm 1 (which we denote **First-order**). We experimented with this modification, as in the authors' experiences, it empirically helps with convergence for some problems.

The **Lower-bound** lines in Figure 3 correspond to the theoretical convergence rate which we have for projected sub-gradient methods. The Lipschitz constant of the minimum eigenvalue function $X \mapsto \lambda_{\min}(X)$ is 1. We do not estimate the diameter. Instead, we plug in a crude lower-bound of $R = 0.2$ (the actual diameter is much larger, and scales with the dimension of the problem). For the **Lower-bound** curve on the bottom, we directly apply Theorem 3.2.2 of [Nes98]. For the curve on the top, we apply Proposition 4.

In practice, we know that sub-gradient methods can be quite slow to converge (since it takes $\mathcal{O}\left(1/\epsilon^2\right)$ steps to achieve ϵ or less error). Unfortunately, it appears empirically that the eigenvalue problem in Renegar's method lies right on the boundary of the convergence analysis. A more rigorous, theoretical analysis is interesting future work.

7.2 Max-cut

In this section, we benchmark various algorithms on the max-cut SDP. We use various graphs from the Gset[4] dataset, a common benchmark for testing max-cut SDPs. The specific graphs we use are described in Table 1.

Algorithms. We benchmark the following algorithms:

(a) **LR-s**. This is the low-rank factorization method described in Section 5. Here, we set $r = 2\log_2|V|$. In our implementation, we follow the penalty schedule outlined in Section 2.2, with $\gamma = 10$ and $\eta = 1/4$. To optimize L with the penalty σ fixed, we use limited memory BFGS (L-BFGS), setting the termination condition to be when the norm of the gradient is less than 10^{-2} for $\{\text{G10}, \text{G44}, \text{G60}\}$ and 10^{-1} for the remainder of the graphs. Note that Section 4.2 of [BM03b] shows how to change the max-cut problem to an unconstrained low rank factorization at the cost of further non-convexity; we do not evaluate this specialization.

[4]Dataset available for download here: http://web.stanford.edu/~yyye/yyye/Gset/

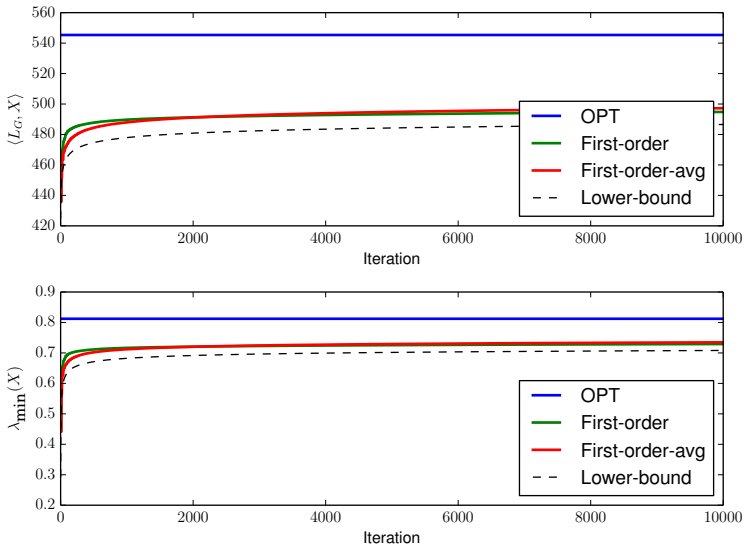

Figure 3: Characteristic performance of Renegar's method on the max-cut problem for random graph instances of $|V| = 50$. The top graph shows the convergence of the method on the original problem, and the bottom graph shows the convergence on the eigenvalue problem. This particular instance was run with stepsize $\eta_t = 1/t$.

Name	$\lvert V\rvert$	$\lvert E\rvert$	Sparsity ($\lvert E\rvert / \lvert V\rvert^2$)
G10	800	19176	0.02996
G44	1000	9990	0.00999
G60	7000	17148	0.00035
G65	8000	16000	0.00025
G77	14000	28000	0.0001429
G81	20000	40000	0.0001

Table 1: Description of the datasets used from the Gset benchmark.

(b) **LR-l**. Same as **LR-s**, except we set $r = \sqrt{|V|}$. Note, from Corollary 2, this is on the same order to guarantee the exact solution is recovered.

(c) **RBR**. This is the specialized max-cut row-by-row block coordinate descent method described in Section 6.2.

(d) **Dual**. This is the unconstrained dual ascent method described in Section 4.3, for SDPs where primal feasible points have constant trace (such as in the case of max-cut).

(e) **Mosek**. This refers to passing the max-cut SDP formulation of (3) to Mosek. We only make this comparison for graph G10 (recall the scaling behavior shown in Figure 1).

Results. Our results are shown in Figure 4. Note that, because the dual problem is unconstrained (hence always feasible), it certifies an upper bound on the optimal value at every iteration. Fur-

thermore, it is not hard to see that every iteration of the RBR method is primal feasible, since the i-th diagonal element is only updated when the i-th row/column is updated. Hence, every iteration of RBR certifies a lower bound on the optimal value; the optimal value must lie between the two curves. The LR method, on the other hand, does not have this sort of guarantee. In Figure 4, we see that initially, when the penalty parameter σ is low, the LR method will move to infeasible points with much larger objective values. Then, as we increase σ, we correct for this infeasibility, and converge towards the actual solution.

The lines in Figure 4 which do not start at time $t = 0$ is an artifact of this experiment, and represents a lack of data about the experiment. More specifically, for LR, this is the time spent in e.g. line search and Hessian approximations in L-BFGS where our code is not called. For Mosek, this is the time spent before the Mosek runtime reports the first interior-point iteration has finished.

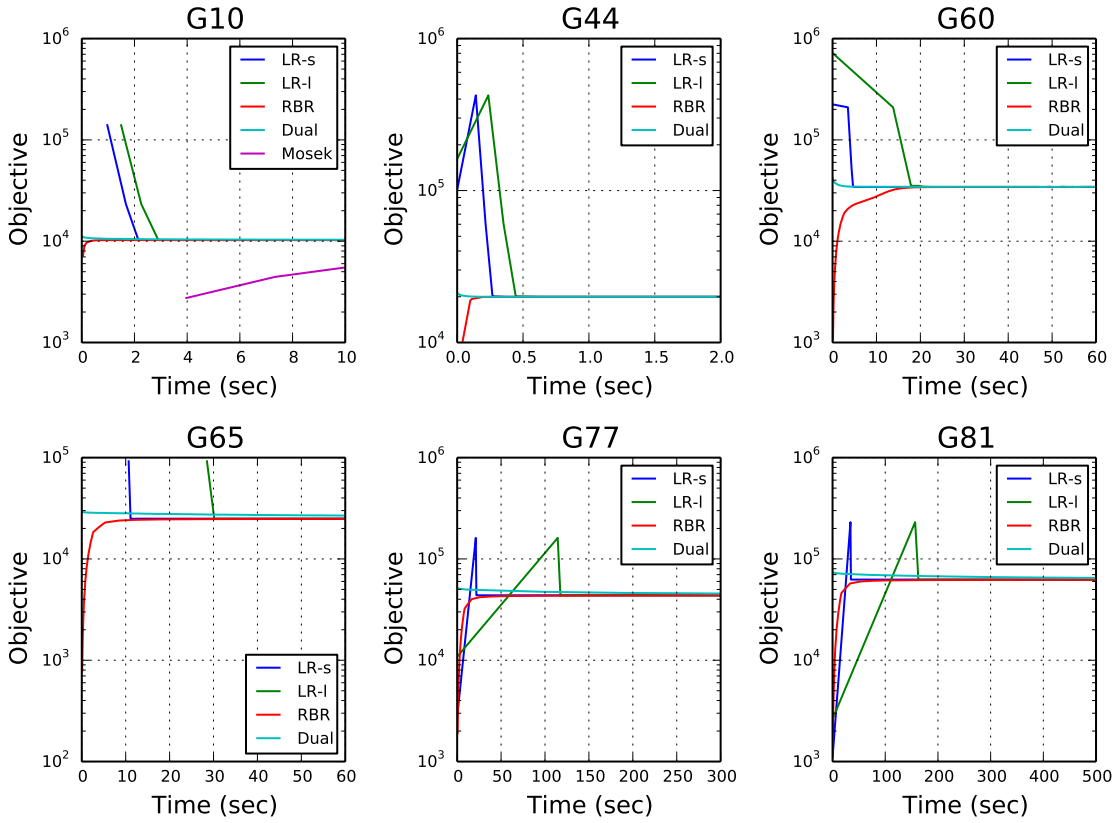

Figure 4: Comparison of running time versus objective value for various SDP algorithms on the max-cut SDP problem for a few graphs from the Gset dataset.

7.3 Matrix completion

In this section, we evaluate both the low-rank factorization method (**LR**) and the specialized row-by-row method (**RBR**) from Section 6.3 on matrix completion problems. We do not include an evaluation of the dual ascent method of multipliers approach outlined in Section 4.2, since we found

empirically that, even with smoothing, the square eigenvalue term makes the function quite difficult to optimize numerically.

7.3.1 Image reconstruction

We first demonstrate these methods visually on an image reconstruction problem. Figure 5 shows the original $m \times n = 128 \times 128$ grayscale image on the left, and the samples we provided the algorithm on the right (the blacked out pixels are unknown to the algorithm). Note that the rank of the original matrix is 96, making the image nearly full rank. Hence, we provide $\mathcal{O}(mn)$ samples to the algorithm to give it a viable chance of recovery [5].

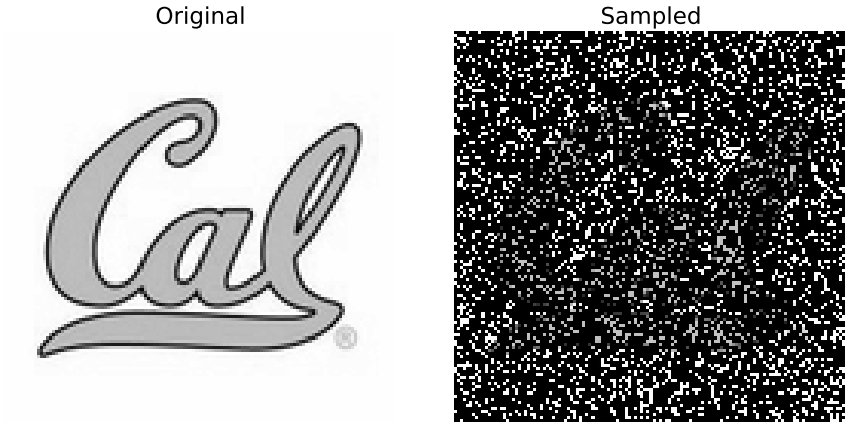

Figure 5: An image reconstruction problem.

We run both the **LR** and **RBR** algorithms on the data shown in Figure 5. Note for **LR**, we set $r = \sqrt{m+n}$. We show the recovered image after each of the first three iteration in Figure 6. We do this to show how the algorithm leaves its initial configuration. Notice that, unlike in the max-cut case, neither algorithm has any guarantee of feasibility at a particular iteration, since both are method of multiplier approaches. Here, we see that after the first iteration, both methods produce a rather infeasible matrix. It is only after the penalty parameter is increased in subsequent iterations that the matrix starts to violate less feasibility constraints.

[5] One might wonder why we chose such a high rank image to demonstrate the matrix completion problem. We were inspired by the authors of [RFP10], who use the MIT logo in their example which happens to be rank 5. It turns out, however, that the Cal logo is of significantly larger rank!

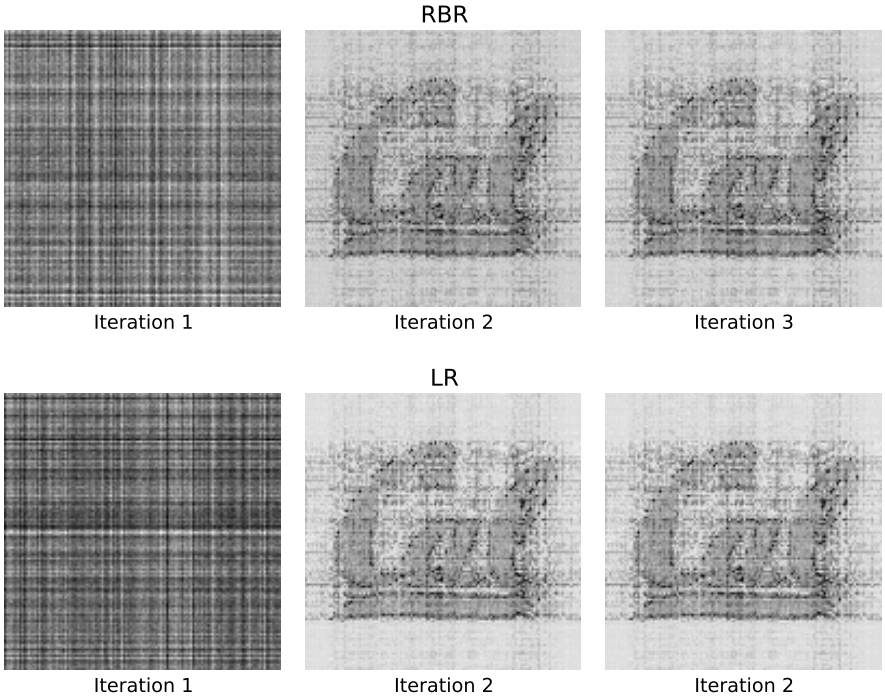

Figure 6: Progress in the first three iterations of **RBR** and **LR** on the image reconstruction matrix completion problem shown in Figure 5.

7.3.2 Random matrix recovery

In this section, we generate random $n \times n$ matrices of rank at most $r = 10$, for $n \in \{500, 1000, 2000\}$. We then sample $2rn$ points uniformly at random, without replacement, from the matrix and run the corresponding matrix completion problem on **LR** and **RBR**. From Theorem 1.1 of [Rec11], $2rn$ samples is on the order of magnitude to guarantee full recovery with high probability (we are mostly ignoring matrix incoherence factors). The results are shown in Figure 7. As in the previous experiment, we set $r = \sqrt{2n}$, which we label as **LR-l** in the figure. Here, we see that, unlike in the max-cut experiment in Figure 4, the **LR** method converges substantially faster than the **RBR** method. This is, however, not surprising. Recall that the specialized RBR algorithm for max-cut is extremely simple, whereas the specialized RBR algorithm for matrix completion requires solving a linear system of equations for every row. Hence, it is comparatively more expensive.

8 Related work

As semidefinite programming has a rich literature surrounding it, it is impossible for a single survey to be fully comprehensive. Here, we highlight a few first order methods which we did not expand on in the previous sections.

Other augmented Lagrangian approaches. Many other authors have proposed an augmented Lagrangian approach to (1). The authors of [WGY10] propose a decomposition which is inspired by ADMM. Unfortunately, each iteration requires a full eigen-decomposition of one of the auxiliary

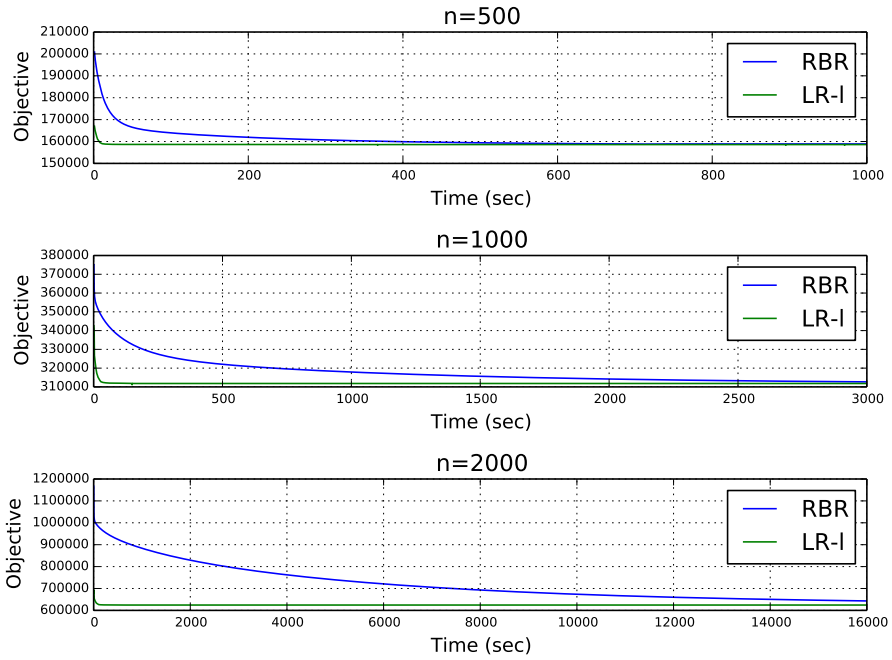

Figure 7: Comparison of the **RBR** method with the **LR** method on random matrix completion problems.

dual variables, which may be prohibitive. However, this method has slightly more promising theory: the authors are able to show that each iteration is contractive for a dual variable (see [WGY10], Lemmas 3-4). Thus, further investigation may be able to uncover finite time convergence rates.

Unconstrained eigenvalue optimization. The eigenvalue problem presented in (13) has sparked independent interest in the research community (with semidefinite programming as an application).

Nesterov in [Nes04] suggests to smooth the eigenvalue problem with the smooth approximation of Lemma 1. Using an accelerated gradient method, we can then recover a convergence rate of $\mathcal{O}\left(1/\sqrt{\epsilon}\right)$ for an ϵ approximation (see e.g. [Nes98]). However, as Lemma 1 suggests, computing the gradient of the smooth approximation requires an eigen-decomposition.

An alternative to smooth approximations is to use cutting plane techniques. At a high level, a cutting plane technique takes advantage of the fact that the sub-gradient at every point defines a hyperplane which is a lower bound on the objective function (this is called a cutting plane). Thus, by keeping multiple sub-gradients around (from past iterations), it is possible to construct an increasingly accurate lower bound estimate for the objective function. Of course, keeping every sub-gradient around is expensive. Thus, spectral bundle methods try to keep only the sub-gradients around which capture most of the approximation. See e.g. [HR00, HK99] for more details.

To address these issues in both smooth approximations and cutting plane methods, the authors of [dEK14] propose to use randomness as the smoothing mechanism. They replace $\lambda_1(X)$ with the

following smooth approximation

$$F_k(X) \stackrel{\text{def}}{=} \mathbb{E}\left[\max_{i=1,\dots,k} \lambda_1(X + \frac{\epsilon}{n} z_i z_i^T)\right]$$

where the z_i's are iid from $\mathcal{N}(0, I_n)$, $\epsilon \geq 0$ is a fixed parameter and $k > 0$ is a fixed constant. It is not hard to see (c.f. [dEK14], Lemma 2.1) that

$$\lambda_1(X) + \frac{\epsilon}{n} \leq F_k(X) \leq \lambda_1(X) + k\epsilon$$

where the lower bound follows from Jensen's inequality and the upper bound follows from the sub-additivity of $\lambda_1(\cdot)$. It turns out that the following procedure produces an unbiased estimator for the gradient $\nabla F_k(X)$. Repeat for $j = 1, ..., q$ the following

(1) Draw $z_i \sim \mathcal{N}(0, I_n)$, $i = 1, ..., k$.

(2) $i_0 \leftarrow \text{argmin}_{i=1,\dots,k} \lambda_1(X + \frac{\epsilon}{n} z_i z_i^T)$.

(3) Let v_j be a normalized eigenvector associated to the top eigenvalue of $X + \frac{\epsilon}{n} z_{i_0} z_{i_0}^T$.

Then use $\frac{1}{q}\sum_{j=1}^q v_j v_j^T$ as the estimate. This procedure is cheaper than doing a full eigen-decomposition of X, and we can apply the theory of stochastic optimization to yield convergence rates.

Acknowledgements

The authors would like to thank Laurent El Ghaoui for his support on this paper and Ben Recht for many fruitful discussions regarding semidefinite programming.

References

[AHO98] Farid Alizadeh, Jean-Pierre A. Haeberly, and Michael L. Overton. Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results. *SIAM J. Optimization*, 8(3), 1998.

[Ber96] Dimitri P. Bertsekas. *Constrained optimization and lagrange multiplier methods*. 1996.

[BLNZ95] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5), 1995.

[BM03a] Samuel Burer and Renato D. C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3), 2003.

[BM03b] Samuel Burer and Renato D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2), 2003.

[BPT12] Grigoriy Blekherman, Pablo A. Parrilo, and Rekha R. Thomas. *Semidefinite Optimization and Convex Algebraic Geometry*. Society for Industrial and Applied Mathematics, 2012.

[CR12] Emmanuel Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6), 2012.

[dEK14] A. d'Aspremont and N. El Karoui. A stochastic smoothing algorithm for semidefinite programming. *SIAM J. Optimization*, 24(3), 2014.

[FXY13] Jiashi Feng, Huan Xu, and Shuicheng Yan. Online robust pca via stochastic optimization. In *NIPS*, 2013.

[Gu97] Ming Gu. Primal-dual interior point methods for semidefinite programming in finite precision. *SIAM J. Optimization*, 10(2), 1997.

[GW94] Michel X. Goemans and David P. Williamson. .879 approximationn algorithms for max cut and max 2sat. In *STOC*, 1994.

[HK99] C. Helmberg and K. C. Kiwiel. A spectral bundle method with bounds. *Mathematical Programming*, 93(2), 1999.

[HR00] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM J. Optimization*, 10(3), 2000.

[HRVW94] Christoph Helmberg, Franz Rendl, Robert J. Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. Technical report, Program in Statistics and Operations Research, Princeton University, 1994.

[JNS13] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *STOC*, 2013.

[KSH97] M. Kojima, S. Shindoh, and S. Hara. Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. *SIAM J. Optimization*, 7(1), 1997.

[MNP98] Pablo Moscato, Michael G. Norman, and Gabor Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Math. Oper. Res.*, 23(2), 1998.

[Nes98] Yuri Nesterov. *Introductory lectures on Convex Programming.* 1998.

[Nes04] Yuri Nesterov. Smoothing technique and its application in semidefinite optimization. *CORE Discussion Paper*, 2004.

[NJS13] Praneeth Netrapalli, Prateek Jain, and Sujay Sanghavi. Phase retrieval using alternating minimization. In *NIPS*, 2013.

[NNS+14] Praneeth Netrapalli, U Niranjan, Sujay Sanghavi, Animashree Anandkumar, and Prateek Jain. Non-convex robust pca. In *NIPS*, 2014.

[PY88] Christos Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *STOC*, 1988.

[Rec11] Benjamin Recht. A simpler approach to matrix completion. *J. Mach. Learn. Res.*, 12, 2011.

[Ren14] J. Renegar. Efficient First-Order Methods for Linear Programming and Semidefinite Programming. *arXiv/1409.5832*, 2014.

[RFP10] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3), 2010.

[VB94] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38(1), 1994.

[WGMS09] Zaiwen Wen, Donald Goldfarb, Shiqian Ma, and Katya Scheinberg. Row by row methods for semidefinite programming. Technical report, Department of IEOR, Columbia University, 2009.

[WGY10] Zaiwen Wen, Donald Goldfarb, and Wotao Yin. Alternating direction augmented lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3-4), 2010.

[Zha05] Fuzhen Zhang. *The Schur Complement and its Applications.* 2005.