Conic Descent Redux for Memory-Efficient Optimization

Bingcong Li, and Georgios B. Giannakis

***Abstract*—Conic programming has well-documented merits in a gamut of signal processing and machine learning tasks. This contribution revisits a recently developed first-order conic descent (CD) solver, and advances it in three aspects: intuition, theory, and algorithmic implementation. It is found that CD can afford an intuitive geometric derivation that originates from the dual problem. This opens the door to novel algorithmic designs, with a momentum variant of CD, momentum conic descent (MOCO) exemplified. Diving deeper into the dual behavior CD and MOCO reveals: i) an analytically justified stopping criterion; and, ii) the potential to design preconditioners to speed up dual convergence. Lastly, to scale semidefinite programming (SDP) especially for low-rank solutions, a memory efficient MOCO variant is developed and numerically validated.**

I. Introduction

Consider a conic programming setup of the form

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \quad \text{s.t. } \mathbf{x}\in\mathcal{K} \tag{1}$$

where the differentiable objective function f is convex, and $\mathcal{K}$ denotes a convex cone. Conic problems are frequently encountered in machine learning and signal processing, where cones can for instance capture non-negative orthant constraints, second-order cones, positive semidefinite cones, exponential cones, and copositive cones [1], [3], [7]. The generality of conic problems fertilizes a number of application domains, leading to the well-documented success in applications such as community detection, and multi-task learning [9], [10], [14].

This work considers first order methods for solving (1). We will focus on Frank Wolfe (FW) variants [8], [12], [17] since their computationally lightweight subproblems can avoid projection onto cones. Taking positive semidefinite cones $\mathbb{S}_+^n$ as an example, projection requires a full SVD with complexity $\mathcal{O}(n^3)$, while a FW subproblem only needs to find out the eigenvector associated with largest eigenvalue for certain matrix, reducing the overall complexity to $\mathcal{O}(n^2)$.

Nonetheless, the noncompact cone constraint prevents applying FW directly on problem (1). A straightforward approach is to include a manually designed constraint to shrink the original constraint set to a compact one $\tilde{\mathcal{K}}$. Consider a simple example with $\mathcal{K} = \{(x,y)|x \geq 0, y \geq 0\}$, one manner to define $\tilde{\mathcal{K}}$ is to turn the non-negative orthant into a polyhedron by adding another constraint, e.g., $x+y\leq 1$. This idea is formalized and generalized in [11], yet prior knowledge is of critical importance to the shrunk constraint $\tilde{\mathcal{K}}$ otherwise it may not contain optimal solutions to (1).

B. Li and G. B. Giannakis are with the Dept. of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA. Emails: {lixx5599, georgios}@umn.edu.

Work [19] considers problem (1) with $\mathcal{K}$ having a relatively simple atomic expression. This additional assumption on $\mathcal{K}$ wipes out constraints such as doubly nonnegative cones, which are useful for reformulating combinatorial problems [5]. Moreover, the convergence rate in [19] depends on the geometry of the cone, thus can be challenged by some "illy conditioned" ones.

A recent method [6] develops conic descent (CD) that can cope with general convex cones regardless of the atomic form of $\mathcal{K}$. However, many of first order approaches, including CD, only target at the primal convergence, leaving the dual properties relatively untouched despite conic duality can be informative. In this work, a detailed study is carried out to understand the dual convergence of CD. In particular, we first provide an explanation to CD that is not only geometrically intuitive, but also having matching mathematical support in the dual domain. This explanation brings up opportunities on algorithmic design, and resulting in a new variant of CD, momentum conic descent (MOCO). MOCO is equipped with heavy ball momentum for faster convergence. Then, extensive theoretical analyses on dual domain bring up deeper insights, and a practical stopping criterion to estimate suboptimality.

We then focus on an instance of (1), SDP problems [20], [24] with the goal of improved scalability. The key observation that motivates the study of memory efficient SDP is that many SDP instances are raised up from vector problems. We term such problems as *raised-up SDPs*. Consider a simple quadratic problem $\min_{\mathbf{x}\in\mathbb{R}^d}\|\mathbf{x}\|_2^2$ as an example. Upon letting $\mathbf{X}:=\mathbf{x}\mathbf{x}^\top$, one can rewrite this problem as $\min_{\mathbf{X}}\mathrm{tr}(\mathbf{X})$ s.t. $\mathbf{X}\in\mathbb{S}_d^+$ and $\mathrm{Rank}(\mathbf{X})\leq 1$. Dropping the rank constraint, one ends up with a SDP problem. While the previous example is too simple to visualize the benefit of raising up vector problem to SDPs, often times such a technique is helpful to turn a nonconvex problem into a convex one; see e.g., [25], [26] for more benefits in real-world applications. However, the raised-up SDP is at an obvious cost of increasing storage relative to its vector form. Our goal is to alleviate such a memory issue leveraging the observation that the desirable solution is usually low rank (recall the rank constraint in our toy example). We propose a memory efficient implementation of MOCO, and leverage Burer-Monteiro (BM) factorization heuristic [4] to further enhance its empirical performances.

In succinct form, our contributions are listed as follows.

- ❖ It is found that conic descent (CD) admits a geometrical explanation. Interestingly, the geometry has a rigorous mathematical foundation in the dual domain of (1).
- ❖ A new algorithm is developed based on the geometrical interpretation. The resultant approach, MOCO, improves

2023 57th Asilomar Conference on Signals, Systems, and Computers | 979-8-3503-2574-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/IEEECONF59524.2023.10476894

the convergence rate of CD, and showcases numerical merits on tested problems.

- ❖ Comprehensive analyses to the dual properties are provided for MOCO. It is observed that the primal and dual convergence do not share the same rate, and the dual behavior can be influenced via preconditioning.
- ❖ We further modify MOCO for memory efficiency of large-scale (raised-up) SDPs. BM heuristic is also incorporated into modified MOCO to facilitate numerical performances.

Notational conventions. Bold lowercase (capital) letters denote column vectors (matrices); $\|\cdot\|$ stands for a norm of either a vector or a matrix, whose dual norm is denoted by $\|\cdot\|_*$; and $\langle\cdot,\cdot\rangle$ is the inner product. Given a cone $\mathcal{K}$, its dual cone is written as $\mathcal{K}^*$. For a set $\mathcal{S}$, we let $\text{dist}(\mathbf{x},\mathcal{S})$ and $\text{dist}_*(\mathbf{x},\mathcal{S})$ denote the distance of vector $\mathbf{x}$ to set $\mathcal{S}$ w.r.t. $\|\cdot\|$, and $\|\cdot\|_*$, respectively. We use $\mathbb{S}^n$ for symmetric real matrices, and $\mathbb{S}^n_+$ to denote the semidefinite positive cone, i.e., all symmetric real positive semidefinite matrices of size $n\times n$.

Appendices. Missing proofs can be found in the online version of this work [16].

II. Understanding Conic Descent Geometrically

This section first describes in detail the class of problems that we are interested in, and then exemplifies a 2-dimensional toy problem to unveil the underlying intuition of CD.

A. Basic assumptions

We formally pinpoint problem (1) by mildly confining the class of objective functions.

Assumption 1 (Lipschitz continuous gradient)**.** *The objective function $f:\mathcal{K}\to\mathbb{R}$ has L-Lipchitz continuous gradients; i.e., $\|\nabla f(\mathbf{x})-\nabla f(\mathbf{y})\|_*\le L\|\mathbf{x}-\mathbf{y}\|, \forall\,\mathbf{x},\mathbf{y}\in\mathcal{K}$.*

Assumption 2 (Strictly convex loss)**.** *The objective function $f:\mathcal{K}\to\mathbb{R}$ is strictly convex; that is, $f(\mathbf{y})-f(\mathbf{x})>\langle\nabla f(\mathbf{x}),\mathbf{y}-\mathbf{x}\rangle, \forall\,\mathbf{x},\mathbf{y}\in\mathcal{X}$ where $\mathbf{x}\neq\mathbf{y}$.*

Assumption 1 is standard in optimization literatures [6], [12], [15], [17], [18], [21], [28]. Assumption 2 is slightly stronger than the commonly adopted one that only requires convexity. This is because of the need of a regularity condition on f to ensure the existence of an optimal solution. Although not stated, other works such as [19] also need this regularity conditions. For example, it is impossible to minimize $f(x,y)=-x+y^2$, which is not strictly convex, over the cone $\mathcal{K}:=\{(x,y)|x\ge 0,y\ge 0\}$. Nonetheless, Assumption 2 is easily satisfied in practice, since it covers many prevalent loss functions, for example, squared ℓ_2 loss and logistic loss. Note that Assumption 2 is slightly stringent for SDPs, and we will relax it for a large class of SDPs later in Section IV. It is also possible to regulate f with assumptions other than strictly convex. For example, the work [6] assumes f to have no nonzero direction of recession in $\mathcal{K}$. Despite this assumption is difficult to verify in practice, our results extends to this setting after justifying the notation accordingly.

For the constraint, we also require the cone to be convex, implying convexity of (1).

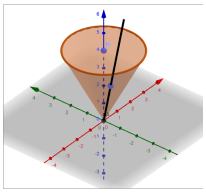

Fig. 1. An example of an ice-cream cone.

Assumption 3 (Convex cone)**.** *The constraint set $\mathcal{K}\in\mathbb{R}^d$ is a convex cone; i.e., $\lambda_1\mathbf{x}+\lambda_2\mathbf{y}\in\mathcal{K}$ for any $\lambda_1\ge 0$, $\lambda_2\ge 0$ and $\mathbf{x},\mathbf{y}\in\mathcal{K}$.*

There are several natural approaches to solve (1) under Assumptions 1 – 3.

Approach 1. Projected gradient descent (GD) is the most straightforward approach. The issue with GD, however, is that projection on a cone can be expensive; see the earlier example of semidefinite positive cone in Section I.

Approach 2. If one has a hint of $\|\mathbf{x}^*\|$, where $\mathbf{x}^*$ is an optimal solution to (1), it is possible to manually impose compactness by including an additional constraint $\|\mathbf{x}\|\le R$ to (1) to clear the obstacles of applying FW. While the FW subproblem is typically much cheaper than projection, a proper estimation on $\|\mathbf{x}^*\|$ is challenging if not impossible. An overestimated $\|\mathbf{x}^*\|$ degrades the performance of FW since its convergence is shaped heavily by the diameter of the constraint [12]; while an underestimated $\|\mathbf{x}^*\|$ may exclude the optimal solution from the feasible domain.

Given the downside of these two approaches, there is a pressing need of more efficient methods. A recent work [6] introduces conic descent (CD). However, the lack of intuition somehow shades the popularity of this approach with great potential. Next, we unveil CD's underlying geometry.

B. Geometric interpretation for CD

Our novel interpretation of CD is built on two key observations. The first one is that a convex cone can be viewed as a set of rotated rays. We will only consider rays initialed at $\mathbf{0}$, that is, $\{t\mathbf{x}|t\ge 0\}$ for some $\mathbf{x}\neq\mathbf{0}$. For example, the first orthant in a $2d$-Cartesian plane can be viewed as the area scanned over by spinning the ray $\{(x,y)|x\ge 0,y=0\}$ counterclockwise by 90 degrees. Another example can be visualized in the ice-cream cone in Figure 1. The second observation is that minimizing over a ray is an $1d$-convex problem and can be solved easily or even analytically.

Indeed, finding an optimal solution $\mathbf{x}^*$ to (1) amounts to seeking for the ray containing it. While it is challenging to find the desirable ray in just a single step, one may progressively improve the *quality* on a ray, which is defined as the minimum function value of this ray. This intuition prompts us to decouple (1) into two (series of) subproblems: i) ray search, where the goal is to guess a ray that may contain $\mathbf{x}^*$; and ii) ray minimization, where this ray is minimized to obtain its quality. The overall goal is that the quality of a ray is improved iteratively until the optimal ray is found. It turns out that CD follows exactly this iterative procedure.

A polar-coordinate perspective. The previous intuition can be understood more concretely through a 2-dimensional example. Consider a simple quadratic objective function

$$f(x,y) = (x-1)^2 + y^2$$

with the cone constraint being the positive orthant, i.e.,

$$\mathcal{K} := \{(x,y) | x \geq 0, y \geq 0\}.$$

This problem can be transformed into polar coordinate, where (x,y) are substituted to angular variables (r,θ), where $r \in [0,+\infty)$, and $\theta \in [0,\frac{\pi}{2}]$. Defining $t := \cos\theta$, we can further change the variables as $x = rt$ and $y = r\sqrt{1-t^2}$. The problem therefore becomes

$$\begin{aligned} \min_{r,t} \ & g(r,t) := (rt-1)^2 + r^2(1-t^2) \\ \text{s.t. } & r \geq 0, \ t \in [0,1]. \end{aligned} \tag{2}$$

From this reformulation (2), it is clear that ray search targets at the optimal t^*, and ray minimization is used to obtain r^* on the previously found ray.

Why not working on polar coordinate. Despite reformulation (2) is geometrically intuitive, challenges remain even for this toy example. The first difficulty comes from the fact that problem (2) is not necessarily convex as it is straightforward to verify that the Hessian of g is negative-definite, i.e.,

$$\nabla^2 g = \begin{bmatrix} 2 & -2 \\ -2 & 0 \end{bmatrix}.$$

Secondly, it is not always easy to reformulate a problem to its polar form, especially for those high dimensional cases. Therefore, it is more attractive to work with non-reformulated form (1), performing ray search in an implicitly manner through the key message from (2), that is, *ray search is essentially a problem on compact domain* ($t \in [0,1]$).

C. The MOCO algorithm

The conic problem (1) can be solved by alternating between ray search and ray minimization as explained in previous subsection. In contrast with CD that adopts vanilla FW for ray search [6], here we propose to augment ray search with with momentum FW [17]. The resultant approach, MOCO, is summarized in Alg. 1. While ray minimizing is straightforward in line 3, ray search is more involved; see lines 4 – 7. Note that MOCO boils down to CD in [6] if $\delta_k \equiv 1$.

It is known that $\nabla f(\mathbf{x}_k)$ is not the best coefficient to use in FW subproblems [17], [18]. This motivates the use of the heavy ball momentum in MOCO. MOCO subproblem in line 5 instead relies on $\mathbf{g}_k$, a weighted average of past gradients. The average $\mathbf{g}_k$ smoothes the possible rapid changes of gradients in consecutive iterations, leading to a more stable searching direction. Another benefit of using momentum is the possibility to continue optimizing even if $\theta_k = 0$. This can be helpful for (matrix SDP) problems with structural solutions, e.g., sparsity or low rankness. The CD iteration stops if $\theta_k = 0$ (since later iterations will not move), and an optimal solution is thus found. On the other hand, the heavy ball momentum further adjusts the weight for $\nabla f(\mathbf{x}_k)$ and

Algorithm 1 Momentum **co**nic descent (MOCO)

1: **Initialize:** $\mathbf{x}_0$, $\delta_k = \frac{2}{k+2} \forall k$
2: **for** $k = 0, 1, \ldots, K$ **do**
3: $\quad \eta_k = \arg\min_{\eta \geq 0} f(\eta \mathbf{x}_k)$ ▷ Ray minimization
4: $\quad \mathbf{g}_k = (1-\delta_k)\mathbf{g}_{k-1} + \delta_k \nabla f(\eta_k \mathbf{x}_k)$
5: $\quad \mathbf{v}_k = \arg\min_{\mathbf{v}} \langle \mathbf{g}_k, \mathbf{v} \rangle$ s.t. $\|\mathbf{v}\| \leq 1, \mathbf{v} \in \mathcal{K}$
6: $\quad \theta_k = \arg\min_{\theta \geq 0} f(\eta_k \mathbf{x}_k + \theta \mathbf{v}_k)$
7: $\quad \mathbf{x}_{k+1} = \eta_k \mathbf{x}_k + \theta_k \mathbf{v}_k$ ▷ Ray search
8: **end for**
9: **Return:** $\eta_K \mathbf{x}_K$

continues optimizing. To see this, note that when $\theta_k = 0$, we have $\eta_{k+1}\mathbf{x}_{k+1} = \eta_k \mathbf{x}_k$. As a result, $\mathbf{g}_{k+2} = (1-\delta_{k+1})\mathbf{g}_{k+1} + \delta_{k+1}\nabla f(\eta_{k+1}\mathbf{x}_{k+1}) = (1-\delta_{k+1})\mathbf{g}_{k+1} + \delta_{k+1}\nabla f(\eta_k \mathbf{x}_k)$, that is, the weight on $\nabla f(\eta_k \mathbf{x}_k)$ is adaptively increased to $\delta_k(1-\delta_{k+1}) + \delta_{k+1}$ if one further unpacks $\mathbf{g}_{k+1}$. This gives a different search direction to continue the search for e.g., solutions with lower rank.

Different from standard FW subproblems, which is $\arg\min_{\mathbf{v} \in \mathcal{K}} \langle \mathbf{g}_k, \mathbf{v} \rangle$, the MOCO subproblem (for ray search) adds an additional constraint $\|\mathbf{v}\| \leq 1$ to ensure that the subproblem is solvable. Concretely, this amounts to our constraint $t \in [0,1]$ in the toy example (2), and the additional constraint can be taken as the range on the cosine of angular variables. Adding the additional constraint $\|\mathbf{v}\| \leq 1$ typically induces no extra computational burden compared to FW subproblems. For example in the SDPs considered later, the subproblems of MOCO and FW have the same complexity. More on ray search will be discussed in Section III-B, where we will view ray search from a duality lens.

III. Primal-dual convergence of MOCO

Having explained the intuition of MOCO, we next focus on its theoretical properties. It is not difficult to see that MOCO converges after the first iteration for any initialization $\mathbf{x}_0 \in \mathcal{K}$ if $\mathbf{x}^* = \mathbf{0}$. We will hence cope with nontrivial problems assuming $\mathbf{x}^* \neq \mathbf{0}$ in the following subsections.

A. Primal convergence

We first deliver a direct result of ray minimization.

Lemma 1. *For every iteration, MOCO ensures that*

$$\langle \eta_k \mathbf{x}_k, \nabla f(\eta_k \mathbf{x}_k) \rangle = 0. \tag{3}$$

Another preparation for the convergence proof is a series of helper functions $\Phi_{k+1}(\mathbf{x})$, defined as

$$\begin{aligned} \Phi_{k+1}(\mathbf{x}) :=& (1-\delta_k)\Phi_k(\mathbf{x}) \\ &+ \delta_k \big[f(\eta_k \mathbf{x}_k) + \langle \nabla f(\eta_k \mathbf{x}_k), \mathbf{x} \rangle \big], \forall k \geq 0. \end{aligned} \tag{4}$$

The definition of $\Phi_0(\mathbf{x})$ does not influence $\Phi_{k+1}(\mathbf{x})$ since $\delta_0 = 0$. Similar to the spirit of [17], the helper functions can be regarded as lower bounds for $f(\mathbf{x})$, where the detailed implications can be found later in Lemma 2. However, there is a key difference that brings up additional challenges to the analysis of MOCO. Unlike [17], $\Phi_{k+1}(\mathbf{x})$ in (4) may have a minimum reaching $-\infty$ due to the noncompactness of

$\mathcal{K}$. Consequently, one cannot adopt $\min_{\mathbf{x}\in\mathcal{K}} \Phi_{k+1}(\mathbf{x})$ directly as the lower bound for $f(\mathbf{x}^*)$. The remedy for this issue is summarized in the next lemma.

Lemma 2. *$\Phi_{k+1}(\mathbf{x})$ satisfies that: i) $\mathbf{v}_k$ minimizes $\Phi_{k+1}(\mathbf{x})$ over $\{\mathbf{x}|\mathbf{x}\in\mathcal{K}, \|\mathbf{x}\|\leq 1\}$; and, ii) there exists $\rho_k \geq 0$ such that $f(\mathbf{x}^*) \geq \Phi_{k+1}(\|\mathbf{x}^*\|\mathbf{v}_k) + \rho_k$ holds, where $\rho_k = 0$ only if $\{\eta_\tau \mathbf{x}_\tau \equiv \mathbf{x}^*\}_{\tau=0}^k$. The rigorous expression of ρ_k can be found in Appendix B.*

Lemma 2 shows that by concentrating on a region that is the intersection of $\mathcal{K}$ and a norm ball, minimizing $\Phi_k(\mathbf{x})$ enables an underestimate of $f(\mathbf{x}^*)$.

Theorem 1 (Primal convergence)**.** *Suppose that Assumptions 1, 2, and 3 hold. Choosing $\delta_k = \frac{2}{k+2}$, MOCO in Alg. 1 guarantees that*

$$f(\eta_{k+1}\mathbf{x}_{k+1}) - \Phi_{k+1}(\|\mathbf{x}^*\|\mathbf{v}_k) \leq \frac{2L\|\mathbf{x}^*\|^2}{k+2} - \rho_k$$

where ρ_k is defined in Lemma 2.

The convergence rate of MOCO can be established as a simple combination of Theorem 1 and Lemma 2.

Corollary 1. *Under assumptions and parameter choices in Theorem 1, Alg. 1 converges with a rate*

$$f(\eta_{k+1}\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \leq \frac{2L\|\mathbf{x}^*\|^2}{k+2} - \rho_k.$$

Comparing the rate of MOCO to its non-momentum counterpart, CD [6], it is observed that momentum tightens the convergence rate by a small term ρ_k. This validates the merits of applying momentum to ray search.

B. Dual convergence

We then tackle the dual convergence of MOCO to gain a complete understanding of its behaviors. Note that our analysis techniques can be directly extended to CD [6].

Definition 1. *Let $\epsilon \geq 0$ be some desirable tolerance. A point $\mathbf{x}_\epsilon^*$ is said to satisfy KKT condition of* (1) *ϵ-approximately if*

$$\mathbf{x}_\epsilon^* \in \mathcal{K} \tag{5a}$$

$$\langle \nabla f(\mathbf{x}_\epsilon^*), \mathbf{x}_\epsilon^* \rangle = 0 \tag{5b}$$

$$\left[dist_*\left(\nabla f(\mathbf{x}_\epsilon^*), \mathcal{K}^*\right)\right]^2 \leq \epsilon. \tag{5c}$$

In particular, (5a) denotes primal feasibility of $\mathbf{x}_\epsilon^*$, (5b) and (5c) characterize complementary slackness and dual feasibility, respectively. Note that the KKT condition is satisfied if $\left[\text{dist}_*\left(\nabla f(\mathbf{x}_\epsilon^*), \mathcal{K}^*\right)\right]^2 = 0$ (i.e., $\epsilon = 0$). Our gaol here is to understand that how fast can $\{\eta_k \mathbf{x}_k\}$ generated by MOCO converge to an ϵ-approximate KKT point.

An obvious fact is that MOCO never generates points outside of $\mathcal{K}$. In other words, (5a) is satisfied automatically. Equation (5b) is also satisfied by $\eta_k \mathbf{x}_k$ as a result of ray minimization; see Lemma 1. This further explains the role of ray minimization, that is, *it seeks $\mathbf{x}^*$ by eliminating points that are not complementarily slack.* It turns out that $\{\eta_k \mathbf{x}_k\}$ is not always dual feasible. Hence, the number of iterations required to ensure dual feasibility characterizes how fast an ϵ-approximate KKT solution is found. Toward this goal, the key inequality leveraged is summarized in the following lemma.

Lemma 3. *Suppose that Assumptions 1 – 2 hold. We have that*

$$f(\mathbf{x}) - f(\mathbf{y}) \geq \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{1}{2L}\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|_*^2.$$

Lemma 3 extends [21, Theorem 2.1.5] to non-Euclidean norms, and it is critical to MOCO's dual convergence.

Theorem 2 (Dual convergence)**.** *Suppose that Assumptions 1, 2 and 3 hold. With $\delta_k = \frac{2}{k+2}$, MOCO guarantees that*

$$\left[dist_*\left(\nabla f(\eta_k \mathbf{x}_k)), \mathcal{K}^*\right)\right]^2 \leq \frac{4L^2\|\mathbf{x}^*\|^2}{k+1}.$$

Theorem 2 asserts that an ϵ-approximate KKT point can be found by MOCO after at most $\mathcal{O}(\frac{L^2\|\mathbf{x}^*\|^2}{\epsilon})$ iterations. A critical observation is that the L dependence is different on primal (Theorem 1) and dual (Theorem 2). This difference will influence the design of stopping criterion, which will be discussed in detail in the upcoming subsection.

C. Stopping criterion

While Theorems 1 and 2 characterize the primal and dual convergence, it is still unclear that when is a good time to stop MOCO iteration. Simply setting $K = \mathcal{O}(\frac{1}{\epsilon})$ works, but it could be too pessimistic since the rates are established for worst cases. In this subsection, we pursue a quantifiable overestimate of suboptimality that not only converges to 0 as k grows, but also can be obtained as a byproduct of MOCO subproblem.

Stopping criterions can be designed based on either primal or dual errors. If working with the primal, $f(\eta_k \mathbf{x}_k) - \Phi_k(\|\mathbf{x}^*\|\mathbf{v}_{k-1})$ in Theorem 1 can be leveraged as an optimality measure. However, its value is impossible to compute due to the lack of knowledge about $\|\mathbf{x}^*\|$. The attempt on dual domain is to rely on $\left[\text{dist}_*\left(\nabla f(\eta_k \mathbf{x}_k), \mathcal{K}^*\right)\right]^2$ in Theorem 2. The issue is, however, computationally expensive and impractical since it requires a projection onto $\mathcal{K}^*$. To overcome these limitations, we find that $\langle \mathbf{g}_k, \mathbf{v}_k \rangle$ approximates $\left[\text{dist}_*\left(\nabla f(\eta_k \mathbf{x}_k), \mathcal{K}^*\right)\right]^2$ well, and can be used as a tractable certification for optimality. To see this, we first write out the dual for MOCO subproblem in line 5,

$$\max_{\mathbf{u}} -\|\mathbf{g}_k - \mathbf{u}\|_* \ \text{ s.t. } \ \mathbf{u} \in \mathcal{K}^*. \tag{6}$$

This dual problem (6) projects $\mathbf{g}_k$ onto $\mathcal{K}^*$, and the optimal objective value is $-\text{dist}_*(\mathbf{g}_k, \mathcal{K}^*)$. Comparing with (5c), it can be seen that long as $\text{dist}_*(\mathbf{g}_k, \mathcal{K}^*) \approx \text{dist}_*(\nabla f(\eta_k \mathbf{x}_k), \mathcal{K}^*)$, one can use the optimal value of (6) as stopping criterion. This observation is formalized in the following theorem.

Theorem 3 (Stopping criterion)**.** *Suppose that Assumptions 1, 2, and 3 hold. Upon choosing $\delta_k = \frac{2}{k+2}$, the following inequality holds for MOCO in Alg. 1 for any $k \geq 2$*

$$\left[dist_*(\mathbf{g}_k, \mathcal{K}^*)\right]^2 \leq \frac{9.7L^2\|\mathbf{x}^*\|^2}{k+1}. \tag{7}$$

Theorem 3 shows that $\text{dist}_*(\mathbf{g}_k, \mathcal{K}^*)$ converges at the same rate of $\text{dist}_*(\nabla f(\eta_k \mathbf{x}_k), \mathcal{K}^*)$ up to constant factors. It further

gives a math interpretation for ray search, that is, *it projects* $\mathbf{g}_k$ *to* $\mathcal{K}^*$ *for (approximated) dual feasibility*.

Next we show that $\text{dist}_*(\nabla f(\eta_k \mathbf{x}_k), \mathcal{K}^*)$ can be conveniently obtained, suiting for the need of the stopping criterion. Strong duality between (6) and line 5 means that $\text{dist}_*(\mathbf{g}_k, \mathcal{K}^*) = -\langle \mathbf{g}_k, \mathbf{v}_k \rangle$. Therefore, one can simply approximate $\text{dist}_*(\nabla f(\eta_k \mathbf{x}_k), \mathcal{K}^*)$ via $\langle \mathbf{g}_k, \mathbf{v}_k \rangle$, and assert an ϵ-approximated KKT point is found whenever

$$\langle \mathbf{g}_k, \mathbf{v}_k \rangle \geq -\mathcal{O}(\sqrt{\epsilon}). \tag{8}$$

It worth pointing out that the criterion (8) is an estimation on dual feasibility, as oppose to the primal error $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ in standard FW literatures [12], [17]. In other words, (8) is no longer affine invariant as in standard FW, opening the possibility for preconditioning.

Preconditioning. With the hope of faster numerical performance, preconditioning applies a linear transformation to $\mathbf{x}$ and solves the transformed problem. It is observed that preconditioning has different impacts on primal and dual of MOCO. In particular, precondition does not reduce $L\|\mathbf{x}^*\|^2$ in primal error (cf. Theorem 1), but it can shrink $L\|\mathbf{x}^*\|$ in dual error (cf. Theorems 2 and 3). Consider the following simple example with $f(x) = (x-2)^2$, whose preconditioned version is given by $g(x) = f(2x) = (2x-2)^2$. We will use subscript f and g to denote constants and variables related to $f(x)$ and $g(x)$, respectively. In this case, one can verify that $L_f\|x_f^*\|_2^2 = L_g\|x_g^*\|_2^2$, but $L_f\|x_f^*\|_2 \neq L_g\|x_g^*\|_2$, demonstrating that the dual error can be scaled down without affecting primal error via proper precondition schemes. Henceforth, an optimal preconditioner can reduce the value of stopping criterion, leading to faster termination of the iterative procedure. This gives new questions on how to find the best preconditioner, which we leave for future work.

IV. Memory efficient MOCO for SDPs

Next, We develop a specific implementation for MOCO to reduce the memory consumption of large-scale semidefinite programing problems (SDP). By leveraging the problem structure, it is possible not only to store vectors in lieu of full matrix variables, but also to relax the regularity condition, i.e., strict convexity, in Assumption 2. We also augment this memory efficient MOCO with a greedy step based on a Burer-Monteiro (BM) factorization heuristic. When injecting a greedy step, it usually improves MOCO convergence.

A. Problem statement

Consider SDPs of the following form

$$\min_{\mathbf{X}} \; f\big(\mathcal{G}(\mathbf{X}) - \mathbf{z}\big) \quad \text{s.t. } \mathbf{X} \in \mathbb{S}_+^n \tag{9}$$

where $\mathbf{z} \in \mathbb{R}^d$ is a given vector. The linear operator $\mathcal{G}$ maps $\mathbf{X} \in \mathbb{S}_+^n$ to $\mathbb{R}^d$, and it is defined as $\mathcal{G}(\mathbf{X}) := [\text{tr}(\mathbf{G}_1\mathbf{X}), \ldots, \text{tr}(\mathbf{G}_d\mathbf{X})]^\top$, where $\mathbf{G}_i \in \mathbb{S}^n, i = 1, \ldots, d$. Problem (9) appears frequently in machine learning and statistics, where $\{\mathbf{G}_i\}$ are often structural, e.g., low rank, sparse, or discrete Fourier transformation matrices. Given $\mathcal{G}$, its adjoint on a vector $\mathbf{a} \in \mathbb{R}^d$ is

$$\mathcal{G}^*(\mathbf{a}) = a_1\mathbf{G}_1 + \ldots + a_d\mathbf{G}_d. \tag{10}$$

In the sequel, we assume that $n^2 \gg d$, and efficient methods exist for computing matrix-vector product $\mathbf{G}_i\mathbf{v}, \forall i$. The latter can be easily satisfied relying on the inherit structure of $\mathbf{G}_i$.

For notational convenience, we will write $f \circ \mathcal{G}(\mathbf{X}) := f\big(\mathcal{G}(\mathbf{X}) - \mathbf{z}\big)$, where $f : \text{dom } f \in \mathbb{R}^d \mapsto \mathbb{R}$, and $f \circ \mathcal{G} : \mathbb{S}_+^n \mapsto \mathbb{R}$. The matrix norm $\|\cdot\|$ in this section refers to Schatten 1-norm (also known as nuclear or trace norm), and its dual norm, $\|\cdot\|_\infty$, is therefore the Schatten-inf norm (or operator norm). The inner product of matrices is standard trace inner product. We do not strict the form for vector norm.

B. Memory efficient implementation of MOCO

Applying MOCO for solving (9) requires the storage of $n \times n$ matrices $\mathbf{X}_k$ and $\mathbf{g}_k$. Note that here $\mathbf{g}_k$ is a matrix, and we keep the same notation as Alg. 1 for consistence. This memory consumption is a significant barrier for scaling problems up. Moreover, it is extremely not economical for raised-up SDPs as discussed in Section I. To facilitate memory efficiency in MOCO, the changes are represented below.

Vectorized representation of $\mathbf{X}_k$. Let $\mathbf{y}_k = \mathcal{G}(\mathbf{X}_k) - \mathbf{z}$. The vector $\mathbf{y}_k$ is helpful for memory saving of MOCO iterates. In particular, η_k can be obtained using only vectors $\mathbf{y}_k$ and $\mathbf{z}$

$$\begin{aligned}\eta_k &= \arg\min_{\eta \geq 0} f \circ \mathcal{G}(\eta \mathbf{X}_k) = \arg\min_{\eta \geq 0} f\big(\mathcal{G}(\eta \mathbf{X}_k) - \mathbf{z}\big) \\ &= \arg\min_{\eta \geq 0} f\big(\eta \mathcal{G}(\mathbf{X}_k) - \mathbf{z}\big) = \arg\min_{\eta \geq 0} f(\eta \mathbf{y}_k + \eta \mathbf{z} - \mathbf{z}).\end{aligned}$$

Similarly, $\mathbf{y}_k$ avoids explicit use of $\mathbf{X}_k$ when solving for θ_k in line 6 in Alg. 2. Another merit of $\mathbf{y}_k$ lies in the fact that $\nabla f \circ \mathcal{G}(\mathbf{X}_k) = \mathcal{G}^*\big(\nabla f(\mathbf{y}_k)\big)$. Owing to the linearity of $\mathcal{G}^*$, i.e., $\mathcal{G}^*(\mathbf{a} + \mathbf{b}) = \mathcal{G}^*(\mathbf{a}) + \mathcal{G}^*(\mathbf{b})$, it is possible to leverage a vector $\tilde{\mathbf{g}}_k \in \mathbb{R}^d$ to retrieve the full gradient $\mathbf{g}_k$ as $\mathcal{G}^*(\tilde{\mathbf{g}}_k)$; see line 4 of Alg. 2.

MOCO subproblem. The MOCO subproblem in line 5 under Schatten 1-norm is equivalent to find the minimum eigenvalue and its normalized eigenvector of $\mathcal{G}^*(\tilde{\mathbf{g}}_k)$. This can be carried out efficiently through shifted power method or the Lanczos method [22].

Sketched representation of $\mathbf{X}_k$. Although $\mathbf{y}_k$ removes the explicit need of $\mathbf{X}_k$, it does not support to reconstruct $\mathbf{X}_k$. Random sketches $\mathbf{S}_k$ are adopted to address this problem in memory efficient form following [27]. In particular, a random Gaussian matrix $\mathbf{\Omega} \in \mathbb{R}^{n \times R}$ is fixed for some predefined parameter $R \ll n$. The sketch is then defined as $\mathbf{S}_k = \mathbf{X}_k\mathbf{\Omega}$. The linearity of sketch also permits a simple update for $\mathbf{S}_k$ in line 8 of Alg. 2. For the ease of understanding and analyses, the update of $\mathbf{X}_k$ is written in line 9, however, this line should be omitted when coding. The overall memory consumption for Alg. 2 is $\mathcal{O}(d + nR)$, which can be much less than $\mathcal{O}(n^2)$ in the naive implementation of MOCO.

Recover $\mathbf{X}_k$ from $\mathbf{S}_k$. One can find a rank r approximation $\hat{\mathbf{X}}_k$ to the real variable $\mathbf{X}_k$ using a stable implementation [27, Algorithm 5.1]. The reconstruction error is bounded as the following if $r < R - 1$

$$\mathbb{E}\big[\|\mathbf{X}_a - \hat{\mathbf{X}}_a\|\big] \leq \left(1 + \frac{r}{R - r - 1}\right) \sum_{i=r+1}^{n} \sigma_i(\mathbf{X}_a).$$

Algorithm 2 Memory efficient MOCO for (9)

Initialize: $\mathbf{y}_0 = -\mathbf{z}$, $\delta_k = \frac{2}{k+2} \forall k, \mathbf{S}_0 = \mathbf{0} \in \mathbb{R}^{n\times r}$
for $k = 0, 1, \ldots, K$ **do**
  $\eta_k = \arg\min_{\eta\geq 0} f(\eta \mathbf{y}_k + \eta \mathbf{z} - \mathbf{z})$
  $\tilde{\mathbf{g}}_k = (1-\delta_k)\tilde{\mathbf{g}}_{k-1} + \delta_k \nabla f(\eta \mathbf{y}_k + \eta \mathbf{z} - \mathbf{z})$
  find $\lambda_k = \lambda_{\min}(\mathcal{G}^*[\tilde{\mathbf{g}}_k])$ and associated normalized eigenvector $\mathbf{q}_k$
  $\theta_k = \arg\min_{\theta\geq 0} f(\eta_k \mathbf{y}_k + \eta_k \mathbf{z} - \mathbf{z} + \theta \mathcal{G}(\mathbf{q}_k \mathbf{q}_k^\top))$
  $\mathbf{y}_{k+1} = \eta_k \mathbf{y}_k + \eta_k \mathbf{z} - \mathbf{z} + \theta_k \mathcal{G}(\mathbf{q}_k \mathbf{q}_k^\top)$
  Option I: $\mathbf{S}_{k+1} = \eta_k \mathbf{S}_k + \theta_k \mathbf{q}_k (\mathbf{q}_k^\top \mathbf{\Omega})$
  Option II: $\mathbf{X}_{k+1} = \eta_k \mathbf{X}_k + \theta_k \mathbf{q}_k \mathbf{q}_k^\top$
  (optional) greedy step in Alg. 3
end for
Return: $\eta_K \mathbf{S}_K$ (to recover $\mathbf{X}_K$)

Algorithm 3 Greedy step at iteration k

find $(t_k, \mathbf{U}_k)$ by solving (11)
$\mathbf{y}_{k+1} \leftarrow t_k^2(\mathbf{y}_{k+1} + \mathbf{z}) + \mathcal{G}(\mathbf{U}_k \mathbf{U}_k^\top) - \mathbf{z}$
$\mathbf{S}_{k+1} \leftarrow t_k^2(\mathbf{S}_{k+1} + \mathbf{z}) + \mathbf{U}_k(\mathbf{U}_k^\top \mathbf{\Omega}) - \mathbf{z}$

The reconstruction error is sufficient small when the true $\mathbf{X}_k$ is low rank. This means that the memory efficiency is almost obtained for free for problems such as raised-up SDPs.

C. Convergence

This subsection strengthens Theorems 1 – 3 by relaxing Assumption 2. Due to space limitation, please find the details in Section IV. C of the online version [16].

D. Practical heuristic 1: Greedy step

This heuristic aims to handle raised-up SDPs through Burer-Monteiro (BM) factorization [2], [4]. The idea is to greadily move $\mathbf{X}_{k+1}$ to a point on another ray to reduce the objective value. This can be done by solving the following unconstrained problem through any descent method [23]

$$\begin{aligned}(t_k, \mathbf{U}_k) &= \underset{t\in\mathbb{R}, \mathbf{U}\in\mathbb{R}^{n\times r}}{\arg\min} f\big(\mathcal{G}(t^2 \mathbf{X}_{k+1} + \mathbf{U}\mathbf{U}^\top) - \mathbf{z}\big) \quad (11)\\ &= \underset{t,\mathbf{U}}{\arg\min} f\big(t^2(\mathbf{y}_{k+1} + \mathbf{z}) + \mathcal{G}(\mathbf{U}\mathbf{U}^\top) - \mathbf{z}\big).\end{aligned}$$

Note that $t_k \mathbf{X}_k + \mathbf{U}_k \mathbf{U}_k^\top$ is a positive semidefinite matrix. Then, the feasible point $t_k \mathbf{X}_k + \mathbf{U}_k \mathbf{U}_k^\top$ is used as the starting point of next iteration of Alg. 2. Accordingly, the way to update $\mathbf{y}_k$ and $\mathbf{S}_k$ based on (11) is given in Alg. 3. The greedy step is optional and can be helpful to run every a few (e.g., 100) iterations to speedup convergence.

Another manner to understand the greedy step is by viewing MOCO as a theoretical justified wrapper for the BM approach, where the convergence of latter is difficult to establish. Because problem (11) is solved through a *descent* approach, the greedy step aided MOCO converges naturally. Note that when choosing a proper solver for (11), the memory consumption of the greedy step aided MOCO is still $\mathcal{O}(d + nR)$. Although the greedy step also applies to CD [6], we find that it is more suitable for MOCO because of the improved performance as shown later in our numerical tests.

E. Practical heuristic 2: magical θ_k

Next, we introduce another practical variant when $\|\mathbf{X}^*\|$ can be estimated. This variant can be useful for raised-up SDPs especially when $\mathbf{X}^* = \mathbf{x}^*(\mathbf{x}^*)^\top$ for some vector $\mathbf{x}^*$. In this case, it can be possible to use the relation $\|\mathbf{X}^*\| = \|\mathbf{x}^*\|_2^2$ to estimate $\|\mathbf{X}^*\|$. An example will be provided in Section V-B together with numerical tests.

This heuristic is motivated by the empirical wisdom that line search can be conservative for numerical performances of heavy ball momentum for FW [17]. Let $M > 0$ be an estimate of $\|\mathbf{X}^*\|$, then our heuristic step size is $\theta_k = \frac{2M}{k+2}$. This step size comes from the detailed derivation of Theorem 1; see the first line of Appendix C. For problems where M is difficult to estimate, it is also possible we can run MOCO for a few iterations, then use $\|\mathbf{X}_k\|$ as an estimate of $\|\mathbf{X}^*\|$. The heuristic θ_k eliminates the need for line search, therefore saving runtime.

V. Numerical tests

Experiments are conducted to visualize the performance of the proposed MOCO and its practical heuristics.

A. Matrix completion

MOCO is first tested on matrix completion problems using synthetic data. Suppose the ground truth matrix $\mathbf{A} \in \mathbb{S}_+^n$ to be recovered is low rank and positive semidefinite. We are given noisy entries of $\mathbf{A}$ sampled randomly, that we denote as $b_{ij} = A_{ij} + \epsilon_{ij}$ for some index $(i,j) \in \mathcal{I}$, where ϵ_{ij} are zero mean i.i.d. Gaussian random variables. Let $\mathbf{A} = \mathbf{V}\mathbf{V}^\top$ for some $\mathbf{V} \in \mathbb{R}^{n\times 3}$ denote the low-rank ground truth. The estimated matrix can be found by solving the following problem

$$\min_{\mathbf{X}} \ \frac{1}{2} \sum_{(i,j)\in\mathcal{I}} (X_{ij} - b_{ij})^2 \quad \text{s.t.} \ \ \mathbf{X} \in \mathbb{S}_n^+. \qquad (12)$$

To understand how MOCO scales, we consider (12) with number of data $n \in \{100, 200, 400, 800, 1600\}$. Following [6], we sample every entry in the upper left 10×10 block and other entries with probability 0.1. The Gaussian noise ϵ_{ij} is randomly generated so that the SNR is 20dB.

The benchmark algorithms are chosen as CD and CD with a greedy heuristic (CDg) [6]. Our numerical tests rely on memory efficient implementation of MOCO. MOCO with the greedy heuristic (MOCOg) is also considered to improve numerical performance. Each tested algorithm is run for 300 iterations. The primal error versus runtime is plotted in Figure 2. For the matrix completion problem, MOCO exhibits slightly worse performance compared to CD. On the other hand, the greedy step appears to be more suitable for MOCO since it clearly boosts the performance of MOCO but not CD with the only exception on the test case with smallest scale $n = 100$. In addition, given the same amount of time, MOCOg achieves the lowest primal error compared with other tested algorithms, thus confirming its scalability and efficiency. The greedy step does not make enough progress for CD, but it significantly helps MOCO. This empirically suggests that the merits of the greedy step are amplified by the momentum in MOCO.

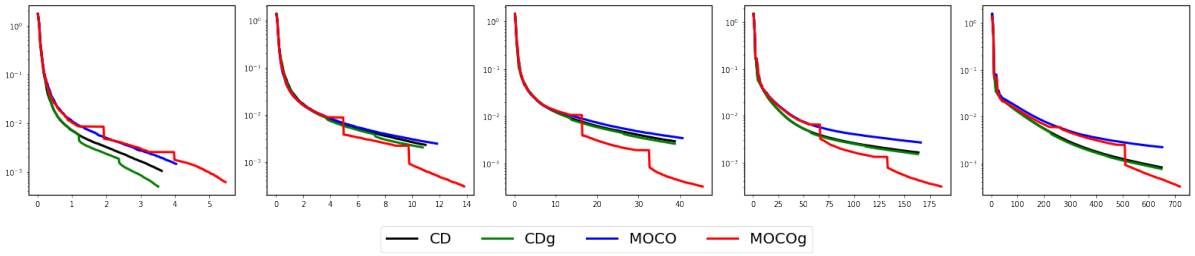

Fig. 2. Performances (runtime vs primal error) of different algorithms for the matrix completion problem (12). From left to right, the sizes of problems are $n = 100, 200, 400, 800, 1600$.

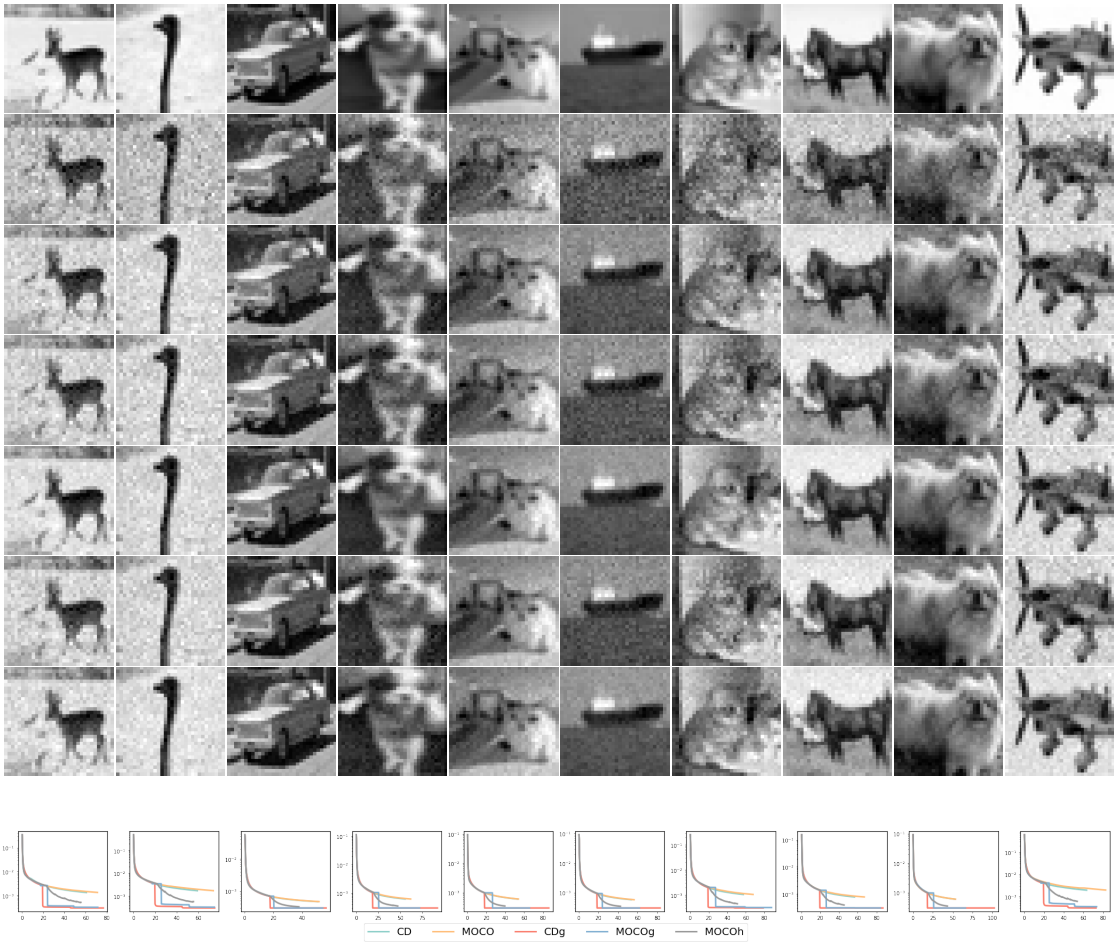

Fig. 3. Performances of various algorithms for the phase retrieval problem. Each column contains the result using a specific image. The first row plots raw images, and other rows (from 2nd to 7th) contain images recovered using FW, CD, MOCO, CDg, MOCOg, MOCOh, respectively. And the last row lists the optimality error vs iteration of compared approaches.

B. Phase retrieval

Suppose that $\mathbf{x} \in \mathbb{R}^n$ is a signal to be retrieved from measurements $b_i = (\mathbf{a}_i^\top \mathbf{x})^2 + \epsilon_i$, where $\mathbf{a}_i \in \mathbb{R}^n$ are rows of matrix $\mathbf{A} = [\mathbf{D}\mathbf{S}_1, \ldots, \mathbf{D}\mathbf{S}_m]^\top$ with $\mathbf{D}$ being the discrete cosine transform and $\mathbf{S}_1, \ldots, \mathbf{S}_m$ being diagonal matrices of independent random signs. One means to recover the original signal is to solve the following problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{mn} \sum_{i=1}^{mn} \|b_i - (\mathbf{a}_i^\top \mathbf{x})^2\|_2^2 + \gamma \|\mathbf{x}\|_2^2.$$

The unsatisfactory of this formulation resides in the fact that the objective function is a polynomial of forth order. This challenges optimization since it is non-smooth. Raised-up SDP is the remedy. Noticing that $(\mathbf{a}_i^\top \mathbf{x})^2 = \mathbf{a}_i^\top \mathbf{x}\mathbf{x}^\top \mathbf{a}_i := \mathbf{a}_i^\top \mathbf{X} \mathbf{a}_i$ where $\mathbf{X} := \mathbf{x}\mathbf{x}^\top$, we can reformulate the problem as

$$\min_{\mathbf{X}\in\mathbb{S}_+^n} \frac{1}{mn} \big\|\mathcal{A}(\mathbf{X}) - \mathbf{b}\big\|^2 + \gamma \operatorname{tr}(\mathbf{X}) \tag{13}$$

where $\mathcal{A} = [\operatorname{tr}(\mathbf{a}_1\mathbf{a}_1^\top \mathbf{X}), \ldots, \operatorname{tr}(\mathbf{a}_{mn}\mathbf{a}_{mn}^\top \mathbf{X})]$. Since $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$, it is natural to assume that there exists a rank-1 optimal

solution $\mathbf{X}^* = \mathbf{x}^*(\mathbf{x}^*)^\top$. The rank-1 assumption also enables an estimate of $\|\mathbf{X}^*\|$ for the heuristic θ_k

$$\|\mathbf{X}^*\| = \mathrm{tr}(\mathbf{X}^*) = \|\mathbf{x}^*\|_2^2 = \frac{1}{m}\sum_{i=1}^{mn}(\mathbf{a}_i^\top \mathbf{x}^*)^2 \approx \frac{1}{m}\sum_{i=1}^{mn} b_i. \quad (14)$$

For the experiment setup, 10 images from CIFAR10 dataset are randomly chosen as the raw signal $\mathbf{x}^*$. The Gaussian noise ϵ_i is generated with 20dB SNR. Other parameters are set to $m = 10$ and $\gamma = 5 \times 10^{-5}$. The benchmark algorithms are chosen as FW, CD, and CD with greedy heuristic (CDg). When working with FW, we add another constraint $\mathrm{tr}(\mathbf{X}) \leq \frac{2}{m}\sum_{i=1}^{mn} b_i$ to ensure the compactness of the constraint set, where the right hand side of this inequality is roughly $2\|\mathbf{X}^*\|$. Three MOCO variants are tested: MOCO in Alg. 2, MOCO with greedy heuristic (MOCOg), and MOCO with heuristic θ_k (MOCOh). All algorithms are run for 300 iterations. We use $R = 3$ for the sketches.

The original and recovered figures are shown in Fig. 3, where the first row lists raw images, and other rows are recovered images via FW, CD, MOCO, CDg, MOCOg, and MOCOh, respectively. Among all implemented approaches, the recovered images using FW have the worst quality. CD and MOCO have almost the same recovery quality, and CDg, MOCOg and MOCOh share the best figure quality. This demonstrates that MOCO and CD not only improve numerical performances over FW, but remove the need for the compact domain requirement.

To further showcase the merits of MOCO over CD, we also plot $f \circ \mathcal{G}(\mathbf{X}_k) - f \circ \mathcal{G}(\mathbf{X}^*)$ versus runtime. The loss curve for FW is omitted here because it works on a different problem from (13) due to the added constraint. Despite the runtime of MOCO is longer than CD because of updating $\tilde{\mathbf{g}}_k$, the runtime of MOCOg is less than that of CDg. Hence, the greedy heuristic is better use with MOCO than CD. Although we do not have an exact explanation, our guess is that the loss curvature of the greedy subproblem could be ill-conditioned in CD. In addition, although relying on a heuristic θ_k, MOCOh often converges faster than MOCO, and even matches to the performance of MOCOg sometimes. As the heuristic θ_k eliminates the need for line search, the runtime of MOCOh is shorter than MOCO.

VI. Conclusion

This paper revisits conic descent (CD) for conic programming problems. CD is refined through a geometrical interpretation that has matching mathematical foundation in the dual domain. Then a new approach, momentum conic descent (MOCO), is proposed to improve CD empirically and theoretically. Lastly, the dual behavior of MOCO (as well as CD) is comprehensively examined, where it is discusses about stopping criterion and opportunities to accelerate convergence via preconditioning. A memory efficient implementation of MOCO for SDPs is then developed based. Memory efficiency is achieved almost for free given the low rankness of the solution. Numerical results further validate the efficiency of proposed MOCO and its practical variants.

References

[1] "MOSEK modeling cookbook 3.3.0," https://docs.mosek.com/modeling-cookbook/powo.html.

[2] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi, "Dropping convexity for faster semi-definite optimization," in *Conference on Learning Theory*. PMLR, 2016, pp. 530–582.

[3] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[4] S. Burer and R. D. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Mathematical Programming*, vol. 95, no. 2, pp. 329–357, 2003.

[5] Y. Cui, L. Liang, D. Sun, and K.-C. Toh, "Projecting onto the degenerate doubly nonnegative cone," *arXiv preprint arXiv:2009.11272*, 2020.

[6] J. C. Duchi, O. Hinder, A. Naber, and Y. Ye, "Conic descent and its application to memory-efficient optimization over positive semidefinite matrices," *Advances in Neural Information Processing Systems*, vol. 33, pp. 8308–8317, 2020.

[7] M. Dür, "Copositive programming – a survey," in *Recent advances in optimization and its applications in engineering*. Springer, 2010.

[8] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[9] B. Hajek, Y. Wu, and J. Xu, "Achieving exact cluster recovery threshold via semidefinite programming," *IEEE Transactions on Information Theory*, vol. 62, no. 5, pp. 2788–2797, 2016.

[10] G. A. Hanasusanto and D. Kuhn, "Conic programming reformulations of two-stage distributionally robust linear programs over wasserstein balls," *Operations Research*, vol. 66, no. 3, pp. 849–869, 2018.

[11] Z. Harchaoui, A. Juditsky, and A. Nemirovski, "Conditional gradient algorithms for norm-regularized smooth convex optimization," *Mathematical Programming*, vol. 152, no. 1, pp. 75–112, 2015.

[12] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization." in *Proc. Intl. Conf. on Machine Learning*, 2013, pp. 427–435.

[13] S. Kakade, S. Shalev-Shwartz, A. Tewari *et al.*, "On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization."

[14] T. Kato, H. Kashima, M. Sugiyama, and K. Asai, "Multi-task learning via conic programming," *Advances in Neural Information Processing Systems*, vol. 20, 2007.

[15] B. Li, M. Coutino, G. B. Giannakis, and G. Leus, "A momentum-guided Frank-Wolfe algorithm," *IEEE Trans. on Signal Processing*, vol. 69, pp. 3597–3611, 2021.

[16] B. Li and G. B. Giannakis, "Conic descent redux for memory-efficient optimization," *arXiv preprint arXiv:2308.07343*, 2023.

[17] B. Li, A. Sadeghi, and G. Giannakis, "Heavy ball momentum for conditional gradient," *Proc. Advances in Neural Info. Process. Syst.*, vol. 34, 2021.

[18] B. Li, L. Wang, G. B. Giannakis, and Z. Zhao, "Enhancing Frank Wolfe with an extra subproblem," in *Proc. of AAAI Conf. on Artificial Intelligence*, 2021.

[19] F. Locatello, M. Tschannen, G. Rätsch, and M. Jaggi, "Greedy algorithms for cone constrained optimization with convergence guarantees," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[20] Y. Nesterov, H. Wolkowicz, and Y. Ye, "Semidefinite programming relaxations of nonconvex quadratic optimization," in *Handbook of semidefinite programming*. Springer, 2000, pp. 361–419.

[21] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2004, vol. 87.

[22] Y. Saad, *Numerical methods for large eigenvalue problems: revised edition*. SIAM, 2011.

[23] J. R. Shewchuk *et al.*, "An introduction to the conjugate gradient method without the agonizing pain," 1994.

[24] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[25] X. Wang, Y. Zhang, G. B. Giannakis, and S. Hu, "Robust smart-grid-powered cooperative multipoint systems," *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6188–6199, 2015.

[26] W. Yu, W. Rhee, S. Boyd, and J. M. Cioffi, "Iterative water-filling for gaussian vector multiple-access channels," *IEEE Transactions on Information Theory*, vol. 50, no. 1, pp. 145–152, 2004.

[27] A. Yurtsever, J. A. Tropp, O. Fercoq, M. Udell, and V. Cevher, "Scalable semidefinite programming," *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 1, pp. 171–200, 2021.

[28] Y. Zhang, B. Li, and G. B. Giannakis, "Accelerating frank-wolfe with weighted average gradients," 2021, pp. 5529–5533.