A Conditional-Gradient-Based Augmented Lagrangian Framework

Alp Yurtsever [1] **Olivier Fercoq** [2] **Volkan Cevher** [1]

Abstract

This paper considers a generic convex minimization template with affine constraints over a compact domain, which covers key semidefinite programming applications. The existing conditional gradient methods either do not apply to our template or are too slow in practice. To this end, we propose a new conditional gradient method, based on a unified treatment of smoothing and augmented Lagrangian frameworks. The proposed method maintains favorable properties of the classical conditional gradient method, such as cheap *linear minimization oracle* calls and *sparse representation* of the decision variable. We prove $\mathcal{O}(1/\sqrt{k})$ convergence rate for our method in the objective residual and the feasibility gap. This rate is essentially the same as the state of the art CG-type methods for our problem template, but the proposed method is arguably superior in practice compared to existing methods in various applications.

1. Introduction

In this paper we focus on the following constrained convex minimization template:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) + g(Bx) \\ \text{subject to} \quad & x \in \mathcal{X} \;\; \& \;\; Ax \in \mathcal{K} \end{aligned} \tag{P}$$

where x is the decision variable that lives on the convex and compact optimization domain $\mathcal{X} \subset \mathbb{R}^n$ with diameter $D_{\mathcal{X}} := \max_{x_1,x_2 \in \mathcal{X}} \|x_1 - x_2\|$; $f : \mathcal{X} \to \mathbb{R}$ is a convex differentiable function with L_f-Lipschitz continuous gradient; $A : \mathcal{X} \to \mathbb{R}^p$ and $B : \mathcal{X} \to \mathbb{R}^q$ are known linear maps; $g : \mathbb{R}^q \to \mathbb{R}$ is a convex function which can be non-smooth but we assume that it is L_g-Lipchitz continuous; $\mathcal{K} \subseteq \mathbb{R}^p$ is a convex set.

[1]LIONS, Ecole Polytechnique Fédérale de Lausanne, Switzerland [2]LTCI, Télécom ParisTech, Université Paris-Saclay, France. Correspondence to: Alp Yurtsever <alp.yurtsever@epfl.ch>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

This template has a large number of applications in machine learning, signal processing, and computer science; from unsupervised clustering (Peng and Wei, 2007) to generalized eigenvector problems (Boumal et al., 2018), and from maximum cut (Goemans and Williamson, 1995) to phase-retrieval (Candès et al., 2013). We refer the reader to Section 5 from (Yurtsever et al., 2018) for a detailed discussion on the special instances and applications of (P).

The conditional gradient method (CGM, *a.k.a.* Frank-Wolfe algorithm) is one of the most scalable methods in the literature for solving convex optimization problems over a structured domain (Yurtsever et al., 2017). The main computational efficiency of CGM comes from the so-called linear minimization oracles (lmo), the main building blocks of CGM:

$$\mathrm{lmo}_{\mathcal{X}}(v) = \arg\min_{x \in \mathcal{X}} \langle x,\, v \rangle.$$

lmo is significantly less expensive than the projection. For instance, lmo contains a rank-1 solution when $\mathcal{X}$ is a nuclear norm-ball, which can be efficiently approximated via Krylov subspace methods. Recall that the projection oracle requires a full singular value decomposition instead.

The classical CGM is originated by Frank and Wolfe (1956), but its resurgence in machine learning follows Hazan and Kale (2012) and Jaggi (2013). Unfortunately, CGM has restrictive assumptions such as the smoothness of the objective function. Therefore, extending CG-type methods for broader templates is an active research area (see Section 4 for some recent advancements). In this work, we introduce the conditional gradient augmented Lagrangian framework (CGAL) for solving (P).

(P) is significantly broader in applications in comparison with the classical CGM template. We can consider the non-smooth term $g(Bx)$ as a regularizer to promote some known structures of the solution, or directly as a non-smooth loss function (*e.g.,* least absolute deviations) for robust optimization. More importantly, (P) contains an affine inclusion constraint $Ax \in \mathcal{K}$. In particular, it covers the standard semidefinite programming (SDP) template (with bounded trace constraint).

Affine constraints are key for the flexibility of the template, but they pose substantial computational difficulty in solving the problem in the primal domain. As a result, primal-

dual methods are typically preferred for solving these problems in large-scale. Among the primal-dual approaches, augmented Lagrangian provides a powerful framework for deriving fast methods. However, the majority of the primal-dual methods for solving (P) rely on the projection and/or proximal-oracles, which do not scale well in many applications (SDP's in particular) and impose a computational bottleneck. In contrast, CGAL provides the utmost scalability by exploiting the *cheap* linear minimization oracles.

CGAL can be viewed as an extension of the recent work of (Yurtsever et al., 2018), from the quadratic penalty to an augmented Lagrangian formulation in the spirit of (Bertsekas, 1976), with the focus on improving its empirical performance. CGAL guarantees $\mathcal{O}(1/\sqrt{k})$ convergence rate both in the objective residual and the feasibility gap. The simplicity of our analysis enables us to identify adaptive bounds, using which we can propose explicit and implementable dual step-size rules that retain the theoretical convergence rates, while significantly enhancing the practical performance of the algorithm. Our numerical evidence demonstrates superior performance.

The rest of this paper is organized as follows. We review the notions of smoothing, quadratic penalty and augmented Lagrangian in Section 2. In Section 3 we introduce CGAL and the main convergence theorem. We provide a detailed discussion of the related works in Section 4. Finally, in Section 5 we present the empirical evidence supporting the advantages of our framework. Technical details are deferred to the supplementary material.

Notation. We use lowercase letters for vectors (or matrices when considering vector space of matrices), uppercase letters for linear maps, and calligraphic letters for sets. We denote the Euclidean inner product by $\langle \cdot, \cdot \rangle$, and the Euclidean norm by $\|\cdot\|$. We denote the adjoint of a linear map A by $A^\top$. For a set $\mathcal{K}$, its indicator function $\iota_{\mathcal{K}} : \mathbb{R}^q \to \mathbb{R} \cup \{+\infty\}$ is defined as

$$\iota_{\mathcal{K}}(z) = \begin{cases} 0 & \text{if } z \in \mathcal{K} \\ +\infty & \text{otherwise.} \end{cases}$$

2. Preliminaries

Our algorithmic design is based on the unified treatment of smoothing, quadratic penalty and augmented Lagrangian frameworks. This section reviews these notions and explains their similarities.

2.1. Nesterov Smoothing

In his seminal work, Nesterov (2005a) introduces a technique for solving some structured non-smooth optimization problems with efficiency estimates $\mathcal{O}(1/\epsilon)$, which is much better than the theoretical lower bound $\mathcal{O}(1/\epsilon^2)$. This technique is known as Nesterov smoothing, and it is commonly used to design efficient primal-dual methods (e.g., (Nesterov, 2005b), (Tran-Dinh et al., 2018)).

Nesterov smoothing exploits an important class of non-smooth functions $\psi(x)$ that can be written in the following *max-form*:

$$\psi(x) = \max_{u \in \mathcal{U}} \left\{ \langle Bx,\, u \rangle - \hat{\phi}(u) \right\},$$

for some convex and compact set $\mathcal{U} \subset \mathbb{R}^q$ and a convex function $\hat{\phi} : \mathcal{U} \to \mathbb{R}$.

Let us consider a prox-function $\delta(u)$ of $\mathcal{U}$, *i.e.,* a strongly convex continuous function on $\mathcal{U}$. Define the center point of this prox-function as

$$\dot{u} = \arg\min_{u \in \mathcal{U}} \delta(u).$$

Without loss of generality, we assume that the strong convexity parameter of δ is 1 and $\delta(\dot{u}) = 0$. Smooth approximation $\psi_\beta(x)$ with the smoothness parameter $\beta > 0$ is defined as

$$\psi_\beta(x) = \max_{u \in \mathcal{U}} \left\{ \langle Bx,\, u \rangle - \hat{\phi}(u) - \beta\delta(u) \right\}.$$

Then, ψ_β is well defined, differentiable, convex and smooth. Moreover, it uniformly approximates ψ, in the sense it satisfies the following envelop property:

$$\psi_\beta(x) \le \psi(x) \le \psi_\beta(x) + \beta D_{\mathcal{U}} \qquad (\forall x \in \mathcal{X}),$$

where $D_{\mathcal{U}} = \max_{u \in \mathcal{U}} \delta(u)$. See Theorem 1 in (Nesterov, 2005a) for the proof and more details.

For notational convenience, we restrict ourselves with $g(B\,\cdot)$, a Lipschitz continuous function coupled with a linear map. Note that we can write $g(B\,\cdot)$ in the max form by choosing $\psi(x) = g(Bx)$ and $\hat{\phi}(u) = g^*(u)$. Here, g^* denotes the Fenchel conjugate of g:

$$g^*(u) = \max_z \left\{ \langle u,\, z \rangle - g(z) \right\}.$$

Since g is convex and lower semicontinuous, Fenchel duality holds, and we have $g(Bx) = g^{**}(Bx)$. Moreover, the Lipschitz continuity assumption of g ensures the boundedness of the dual domain (see Lemma 5 in (Dünner et al., 2016) for a formal statement of this well-known result).

In this work, we specifically focus on the Euclidean prox-functions, $\delta(u) = \frac{1}{2}\|u - \dot{u}\|^2$. By definition of ψ_β, g_β takes the following form:

$$g_\beta(Bx) = \max_{u \in \mathbb{R}^q} \left\{ \langle Bx,\, u \rangle - g^*(u) - \frac{\beta}{2}\|u - \dot{u}\|^2 \right\}.$$

The argument of this maximization subproblem can be written as $\operatorname{prox}_{\beta^{-1} g^*}(\dot{u} + \beta^{-1} Bx)$, where

$$\operatorname{prox}_g(z) = \arg\min_u \; g(u) + \frac{1}{2}\|z - u\|^2.$$

Finally, we can compute the gradient of g_β as

$$\begin{aligned}\nabla g_\beta(Bx) &= B^\top \operatorname{prox}_{\beta^{-1}g^*}(\dot{u} + \beta^{-1}Bx) \\ &= B^\top \dot{u} + \beta^{-1}B^\top \left(Bx - \operatorname{prox}_{\beta g}(\beta\dot{u} + Bx)\right),\end{aligned}$$

where the second line follows from the well-known Moreau decomposition.

2.2. Quadratic Penalty

The quadratic penalty method is an effective proxy for handling the affine constraints $Ax \in \mathcal{K}$. It works by replacing the constraint with the penalty function which favors the feasibility of iterates. We consider the squared Euclidean distance, $\frac{\lambda}{2}\operatorname{dist}^2(Ax, \mathcal{K})$, as the penalty function, and $\lambda > 0$ is called as the penalty parameter. We update this parameter as we progress in the optimization procedure to converge to a solution of the original constrained problem. Surprisingly, the quadratic penalty approach is structurally equivalent to a *de facto* instance of Nesterov smoothing.

Let us start by writing the Fenchel conjugate of the indicator function $\iota_{\mathcal{K}}(\,\cdot\,)$,

$$\iota^*_{\mathcal{K}}(z) = \max_{v\in\mathcal{K}} \langle v,\, z\rangle.$$

Then, we can write the affine constraint in the max form by choosing $\hat{\phi}(z) = \iota^*_{\mathcal{K}}(z)$ and using the following formula:

$$\begin{aligned}\iota_{\mathcal{K}}(Ax) &= \max_z \left\{ \min_{v\in\mathcal{K}} \langle Ax - v,\, z\rangle \right\} \\ &= \max_z \left\{ \langle Ax,\, z\rangle - \iota^*_{\mathcal{K}}(z) \right\}.\end{aligned}$$

By choosing the standard Euclidean prox-function $\delta(v) = \frac{1}{2}\|v\|^2$, we get the following "smooth approximation":

$$\begin{aligned}\iota_{\mathcal{K}\beta}(Ax) &= \max_z \left\{ \min_{v\in\mathcal{K}} \langle Ax - v,\, z\rangle - \frac{\beta}{2}\|z\|^2 \right\} \\ &= \min_{v\in\mathcal{K}} \max_z \left\{ \langle Ax - v,\, z\rangle - \frac{\beta}{2}\|z\|^2 \right\} \\ &= \frac{1}{2\beta}\operatorname{dist}^2(Ax, \mathcal{K}),\end{aligned}$$

where the inversion of $\min$ and $\max$ holds due to the Sion's minimax theorem (Sion, 1958).

In summary, we can obtain the quadratic penalty with parameter $\lambda = \beta^{-1}$, by applying the Nesterov smoothing procedure to the indicator of an affine constraint.

Note that the quadratic penalty does not serve as a uniform approximation, because the dual domain is unbounded and the envelope property does not hold. Consequently, the common analysis techniques for smoothing does not apply for quadratic penalty methods. Nevertheless, one can exploit this structural similarity to design algorithms that universally work for both cases; composite problems with smoothing-friendly non-smooth regularizers, and problems with affine constraints.

Quadratic penalty provides simple algorithms with interpretable steps, but they provide limited practical applicability due to the poor empirical performance. To this end, the next subsection reviews augmented Lagrangian methods as an alternative approach.

2.3. Augmented Lagrangian

Augmented Lagrangian (AL) methods replace the affine constraint with a continuous function that promotes the feasibility, similar to the quadratic penalty approach. This function is parametrized by a penalty parameter $\lambda > 0$ (*aka* augmented Lagrangian parameter) and a dual vector $\dot{v} \in \mathbb{R}^p$ (Lagrange multiplier). In the *min-form*, we can write this function as

$$\min_{v\in\mathcal{K}} \left\{ \langle \dot{v},\, Ax - v\rangle + \frac{\lambda}{2}\|Ax - v\|^2 \right\}.$$

We can view the augmented Lagrangian as a shifted quadratic penalty, since

$$\begin{aligned}&\arg\min_x f(x) + \min_{v\in\mathcal{K}} \left\{ \langle \dot{v},\, Ax - v\rangle + \frac{\lambda}{2}\|Ax - v\|^2 \right\} \\ &\quad= \arg\min_x f(x) + \min_{v\in\mathcal{K}} \frac{\lambda}{2}\|Ax - v + \tfrac{1}{\lambda}\dot{v}\|^2 \\ &\quad= \arg\min_x f(x) + \frac{\lambda}{2}\operatorname{dist}^2(Ax + \tfrac{1}{\lambda}\dot{v}, \mathcal{K}).\end{aligned}$$

Therefore, it is not surprising that we can relate augmented Lagrangian function with Nesterov smoothing. To draw this relation, we simply follow similar arguments as in the quadratic penalty case, but this time we use a shifted prox-function $\delta(v) = \frac{1}{2}\|v - \dot{v}\|^2$:

$$\begin{aligned}\iota_{\mathcal{K}\beta}(Ax) &= \max_z \left\{ \min_{v\in\mathcal{K}} \langle Ax - v,\, z\rangle - \frac{\beta}{2}\|z - \dot{v}\|^2 \right\} \\ &= \min_{v\in\mathcal{K}} \max_z \left\{ \langle Ax - v,\, z\rangle - \frac{\beta}{2}\|z - \dot{v}\|^2 \right\} \\ &= \min_{v\in\mathcal{K}} \left\{ \langle \dot{v},\, Ax - v\rangle + \frac{1}{2\beta}\|Ax - v\|^2 \right\}.\end{aligned}$$

In conclusion, augmented Lagrangian formulation is structurally equivalent to a *de facto* instance of Nesterov smoothing, applied to the indicator of the constraint, with a shifted Euclidean prox-function. The center point of this prox-function corresponds to the dual variable, and the penalty parameter corresponds to the inverse of the smoothness parameter ($\lambda = \beta^{-1}$). Once again, this approach does not serve as a uniform approximation, and the common analysis for Nesterov smoothing does not apply for augmented Lagrangian.

3. Algorithm

In this section, we design CGAL for the special case of $g(Bx) = 0$ for the ease of presentation. One can extend CGAL in a straightforward way for the general case, based on the discussion in Section 2, and the analysis techniques in this work and (Yurtsever et al., 2018).

Algorithm 1 CGAL (for $g(Bx) = 0$)

Input: $x_1 \in \mathcal{X}$, $y_1 \in \mathbb{R}^p$, $\lambda_0 > 0$
for $k = 1, 2, \ldots,$ **do**
$\eta_k = 2/(k+1)$ and $\lambda_k = \lambda_0\sqrt{k+1}$
$r_k = \text{proj}_{\mathcal{K}}\left(Ax_k + (1/\lambda_k)y_k\right)$
$v_k = \nabla f(x_k) + A^\top y_k + \lambda_k A^\top(Ax_k - r_k)$
$s_k = \arg\min_{x\in\mathcal{X}} \langle v_k, x\rangle$
$x_{k+1} = x_k + \eta_k(s_k - x_k)$
$\bar{r}_{k+1} = \text{proj}_{\mathcal{K}}\left(Ax_{k+1} + (1/\lambda_{k+1})y_k\right)$
$\sigma_{k+1} \leftarrow$ using (decr.) or (const.)
$y_{k+1} = y_k + \sigma_{k+1}\left(Ax_{k+1} - \bar{r}_{k+1}\right)$
end for

3.1. Design of CGAL

Let us introduce the slack variable $r = Ax \in \mathcal{K}$ and define the augmented Lagrangian function as

$$\begin{aligned}\mathcal{L}_\lambda(x, y) &= f(x) + \min_{r\in\mathcal{K}}\left\{\langle y,\, Ax - r\rangle + \frac{\lambda}{2}\|Ax - r\|^2\right\}\\ &= f(x) - \frac{1}{2\lambda}\|y\|^2 + \frac{\lambda}{2}\,\text{dist}^2\left(Ax + \frac{1}{\lambda}y, \mathcal{K}\right).\end{aligned}$$

where $y \in \mathbb{R}^p$ is the Lagrange multiplier and $\lambda > 0$ is the penalty parameter. Clearly, $\mathcal{L}_\lambda(x, y)$ is a smooth convex function with respect to x.

One CGAL iteration is composed of three basic steps:

- ▷ Primal step (conditional gradient step on x),
- ▷ Penalty parameter update (increment λ),
- ▷ Dual step (proximal gradient step on y).

Primal step. CGAL is characterized by the conditional gradient step with respect to $\mathcal{L}_\lambda(\,\cdot\,, y)$ on the primal variable. Define

$$r_k = \text{proj}_{\mathcal{K}}\left(Ax_k + \frac{1}{\lambda_k}y_k\right).$$

Then, we can evaluate $\nabla_x\mathcal{L}_{\lambda_k}(x, y_k)$ as

$$\nabla_x\mathcal{L}_{\lambda_k}(x, y_k) = \nabla f(x_k) + A^\top y_k + \lambda_k A^\top(Ax_k - r_k).$$

Next, we query the linear minimization oracle

$$s_k = \arg\min_{x\in\mathcal{X}}\left\langle \nabla_x\mathcal{L}_{\lambda_k}(x_k, y_k),\, x\right\rangle,$$

and we form the next iterate (x_{k+1}) by combining the current iterate x_k and s_k with the CG step-size η_k. We use the classical step size $\eta_k = 2/(k+1)$ of CG-type methods, but the same guarantees hold for the design variants with line-search or fully corrective updates.

Penalty parameter update.

Penalty methods typically require the penalty parameter to be increased at a certain rate for provable convergence. In contrast, augmented Lagrangian methods can be designed with a fixed penalty parameter, because the saddle point formulation already favors the constraints. Unlike other augmented Lagrangian CG-type methods, we adopt an increasing penalty sequence in CGAL by choosing $\lambda_k = \lambda_0\sqrt{k+1}$ for some $\lambda_0 > 0$.

Dual step. Once x_{k+1} is formed, we update the dual variable y_k by a gradient ascent step with respect to $\mathcal{L}_\lambda(x, \cdot)$. At iteration k, we evaluate dual update by

$$y_{k+1} = y_k + \sigma_{k+1}\nabla_y\mathcal{L}_{\lambda_{k+1}}(x_{k+1}, y_k).$$

To compute $\nabla_y\mathcal{L}_{\lambda_{k+1}}$, we first define

$$\bar{r}_{k+1} = \text{proj}_{\mathcal{K}}\big(Ax_{k+1} + \tfrac{1}{\lambda_{k+1}}y_k\big).$$

Then, we can use the following formulation:

$$\nabla_y\mathcal{L}_{\lambda_{k+1}}(x_{k+1}, y_k) = Ax_{k+1} - \bar{r}_{k+1}.$$

The choice of dual step-size is crucial for convergence guarantees. We propose two alternative schemes, with a decreasing or constant bound on the step-size.

Decreasing bound on step-size. This variant cancels positive quadratic terms in the majorization bounds due to dual updates, with the negative quadratic terms that come from the penalty parameter updates. Consequently, we choose the largest $\sigma_{k+1} \geq 0$ which satisfies

$$\sigma_{k+1} \leq \tfrac{\lambda_0}{2\sqrt{k+1}} \quad \& \quad \|y_{k+1}\| \leq D_{\mathcal{Y}_{k+1}} \qquad \text{(decr.)}$$

$D_{\mathcal{Y}_{k+1}}$ is a sequence of positive numbers to be chosen, which acts as a dual domain diameter and appears in the final bounds. We will specify a reasonable positive constant $D_{\mathcal{Y}} = D_{\mathcal{Y}_{k+1}}$ in the sequel from the final converges bounds, by matching the factors of the dominating terms.

Constant bound on step-size. We observed significant performance improvements by slightly relaxing the decreasing upper bound on the step-size. To this end, we design this second variant. We do not cancel out additional quadratic terms but restrict them to be smaller than other dominating terms in the majorization bound. To this end, we choose the largest $\sigma_{k+1} \geq 0$ which satisfies ($D_{\mathcal{Y}_k}$ is similar as in (decr.) case)

$$\begin{aligned}\sigma_{k+1} &\leq \lambda_0\\ \|y_{k+1}\| &\leq D_{\mathcal{Y}_{k+1}} \qquad \text{(const.)}\\ \sigma_{k+1}\|Ax_{k+1} - \bar{r}_{k+1}\|^2 &\leq \tfrac{1}{2}\eta_k^2(L_f + \lambda_{k+1}\|A\|^2)D_{\mathcal{X}}^2.\end{aligned}$$

We underline that the computation of σ_k does not require an iterative line-search procedure. Instead, it can be computed by simple vector operation both in (decr.) and (const.) variants. As a result, the computational cost of finding σ_k is negligible.

3.2. Theoretical Guarantees of CGAL

We present convergence guarantees of CGAL in this section. But first, we define some basic notions to be used in the sequel and state our main assumptions.

Solution set. We denote a solution of (P) by $x^\star$, and the set of all solutions by $\mathcal{X}^\star$. Similarly, we denote a solution of the dual problem by $y^\star$, and the set of all solutions by $\mathcal{Y}^\star$. Throughout, we assume that the solution set is nonempty and that there exists a finite dual solution, *i.e.,* $\min_{y\in\mathcal{Y}^\star}\|y\|<\infty$.

ϵ-solution. Given an accuracy level $\epsilon>0$, we call a point $x\in\mathcal{X}$ as an ϵ-solution of (P) if

$$f(x)-f^\star\leq\epsilon, \quad \text{and} \quad \operatorname{dist}(Ax,\mathcal{K})\leq\epsilon.$$

We call $f(x)-f^\star$ as the objective residual and $\operatorname{dist}(Ax,\mathcal{K})$ as the feasibility gap. Note that the convergence of objective residual alone is not enough to approximate the solution since the iterates are non-feasible and $f(x)-f^\star$ can take negative values.

Strong duality. We assume that the strong duality holds. This assumption is common for primal-dual methods, and the Slater's condition is a widely used sufficient condition for the strong duality:

$$\operatorname{relint}(\mathcal{X}\times\mathcal{K})\cap\{(x,r)\in\operatorname{dom}(f)\times\mathbb{R}^d : Ax=r\}\neq\emptyset,$$

where relint means relative interior.

Theorem 3.1. *Sequence x_k generated by CGAL with dual step-size conditions* (const.) *satisfies:*

$$f(x_k)-f^\star\geq-\|y^\star\|\operatorname{dist}(Ax_k,\mathcal{K})$$

$$f(x_k)-f^\star\leq 4D_{\mathcal{X}}^2\left(\frac{L_f}{k}+\frac{\lambda_0\|A\|^2}{\sqrt{k}}\right)+\frac{D_{\mathcal{Y}_k}^2}{2\lambda_0\sqrt{k}}$$

$$\operatorname{dist}(Ax_k,\mathcal{K})\leq\frac{2/\lambda_0}{\sqrt{k}}\left(\frac{D_{\mathcal{Y}_k}}{2}+\|y_k-y^\star\|+\sqrt{2C_0\lambda_0D_{\mathcal{X}}^2}\right)$$

where $C_0=L_f+\|A\|^2\lambda_0$. We can also bound $\|y_k-y^\star\|$ using triangle inequality. Considering the bounds, it is reasonable to choose $D_{\mathcal{Y}}$ proportional to $D_{\mathcal{X}}\|A\|\lambda_0$.

Sequence x_k generated by CGAL with dual step-size conditions (decr.) *satisfies similar guarantees as* (const.)*, with the factor of $1/2$ for all terms involving $D_{\mathcal{X}}^2$.*

We omit design variants of CGAL with line-search and fully corrective updates, covered by our theory. The same guarantees hold for these variants.

3.3. Extension for Composite Problems

One can extend CGAL in a straightforward way for composite problems based on the discussions in Section 2. For this, we simply need to define the sum of two non-smooth terms: $G(Ax,Bx)=\iota_{\mathcal{K}}(Ax)+g(Bx)$. Then, CGAL guarantees $\mathcal{O}(1/\sqrt{k})$ rates in the feasibility gap $\operatorname{dist}(Ax,\mathcal{K})$ and in the objective residual $f(x)+g(Bx)-f(x^\star)-g(Bx^\star)$. See the supplements for more details.

Below we describe the extension (const.) for this setting

$$\begin{aligned}
&\sigma_{k+1}\leq\lambda_0 \quad \text{and} \quad \gamma_{k+1}\leq\beta_0\\
&\|y_{k+1}\|\leq D_{\mathcal{Y}_{k+1}} \quad \text{and} \quad \|z_{k+1}\|\leq D_{\mathcal{Z}_{k+1}}\\
&\sigma_{k+1}\|Ax_{k+1}-\bar{r}_{k+1}\|^2\leq\tfrac{1}{4}\eta_k^2\bar{\bar{L}}_{k+1}D_{\mathcal{X}}^2\\
&\gamma_{k+1}\|Bx_{k+1}-\bar{t}_{k+1}\|^2\leq\tfrac{1}{4}\eta_k^2\bar{\bar{L}}_{k+1}D_{\mathcal{X}}^2 \qquad \text{(const.2)}
\end{aligned}$$

where $\bar{\bar{L}}_{k+1}=(L_f+\lambda_{k+1}\|A\|^2+\beta_{k+1}^{-1}\|B\|^2)$. A reasonable choice is $D_{\mathcal{Z}_{k+1}}=D_{\mathcal{Z}}=L_g$. One can similarly also extend (decr.) for this setting. Alternatively, we can set $z_k=0_n$ fixed, and perform dual updates only on y_k.

Algorithm 2 CGAL for (P)

Input: $x_1\in\mathcal{X}$, $y_1\in\mathbb{R}^p$, $z_1\in\mathbb{R}^q$, $\lambda_0>0$, $\beta_0>0$
for $k=1,2,\ldots,$ **do**
  $\eta_k=2/(k+1)$, $\lambda_k=\lambda_0\sqrt{k+1}$, $\beta_k=\beta_0/\sqrt{k+1}$
  $r_k=\operatorname{proj}_{\mathcal{K}}(Ax_k+(1/\lambda_k)y_k)$
  $t_k=\operatorname{prox}_{\beta_k g}(Bx_k+\beta_k z_k)$
  $v_k=A^\top y_k+\lambda_k A^\top(Ax_k-r_k)$
  $w_k=B^\top z_k+(1/\beta_k)B^\top(Bx_k-t_k)$
  $s_k=\arg\min_{x\in\mathcal{X}}\langle\nabla f(x_k)+v_k+w_k,x\rangle$
  $x_{k+1}=x_k+\eta_k(s_k-x_k)$
  $\bar{r}_{k+1}=\operatorname{proj}_{\mathcal{K}}(Ax_{k+1}+(1/\lambda_{k+1})y_k)$
  $\sigma_{k+1}\leftarrow$ using (decr.2) or (const.2)
  $y_{k+1}=y_k+\sigma_{k+1}(Ax_{k+1}-\bar{r}_{k+1})$
  $\bar{t}_{k+1}=\operatorname{prox}_{\beta_{k+1}g}(Bx_{k+1}+\beta_{k+1}z_k)$
  $\gamma_{k+1}\leftarrow$ using (decr.2) or (const.2)
  $z_{k+1}=z_k+\gamma_{k+1}(Bx_{k+1}-\bar{t}_{k+1})$
end for

4. Related Work

The majority of convex methods for solving (P) are based on computationally challenging oracles, *e.g.,* some second-order oracle (for interior point methods), the projection onto $\mathcal{X}$ (for operator splitting methods), and a constrained proximal-oracle (for the majority of the classical primal-dual methods). For these methods, we refer to (Wright, 1997), (Komodakis and Pesquet, 2015), (Ryu and Boyd, 2016) and the references therein. In the rest of this section, we focus on the lmo-based algorithms for solving (P) or some special instances of it.

Lan (2014) introduces a conditional gradient method for non-smooth minimization over a convex compact domain.

His method is based on the Nesterov smoothing, and it is the first attempt to combine the Nesterov smoothing and conditional gradient approach, to the best of our knowledge. However, this method does not apply in the presence of affine constraints, since it relies on the boundedness of the dual domain and the uniform approximation property.

Yurtsever et al. (2015) present the universal primal-dual method (UPD), a primal-dual subgradient approach for solving convex minimization problems with affine constraints. The main template of UPD is fairly different than (P); it does not have the non-smooth term $g(Bx)$ and the smoothness assumption on f, but it assumes Hölder smoothness in the dual space instead. The method does not directly work with lmo's, but it leverages the so-called sharp operators. For the standard form SDP formulation, however, the sharp-operator is an instance of the lmo.

UPD adopts the inexact line-search strategy introduced by Nesterov (2015). This strategy requires the target accuracy ϵ as an input parameter, and UPD is guaranteed to converge only up to ϵ accuracy, *i.e.,* UPD guarantees $f(x) - f^\star \leq \mathcal{O}(1/\sqrt{k}) + \epsilon$. The practical performance of UPD heavily depends on this parameter: Choosing ϵ too small leads to very small step-sizes hence slow convergence. Best values for ϵ are typically around $1/10$th and $1/100$th of the $|f^\star|$, but UPD is difficult to tune unless the optimal value is roughly known.

Lan and Zhou (2016) propose the conditional gradient sliding method (CGS). This method is based on an inexact version of the accelerated gradient method by Nesterov (1987), where the projection subproblem is approximately solved by using the classical CGM. CGS is originally proposed for smooth minimization over a convex and compact domain, but the results are generalized for smoothing friendly non-smooth functions in Section 4 by following the same approach as Lan (2014). Note that this generalization directly follows the standard approach of Nesterov smoothing, and it does not apply for affine constraints.

Yen et al. (2016b) propose the greedy direction method of multipliers (GDMM), a CGM variant for minimizing a linear objective function over an intersection of polytopes. GDMM relies on a consensus reformulation over the cartesian product of these polytopes, and the consistency constraint is incorporated by the augmented Lagrangian. This method is further explored in the structural support vector machine (Yen et al., 2016a) and maximum a-posteriori inference (Huang et al., 2017) problems. Nevertheless, Gidel et al. (2018) point out some technical issues in the analysis of this approach, see Section B.1 in (Gidel et al., 2018).

Gidel et al. (2018) propose an augmented Lagrangian framework for the convex splitting problem (FW-AL). Similar to CGAL, this method is characterized by one CGM step on $\mathcal{L}_\lambda(\cdot, y_k)$ followed by one dual gradient ascent step on $\mathcal{L}_\lambda(x_{k+1}, \cdot)$. In contrast to CGAL, the penalty parameter λ of FW-AL is kept fixed. Originally, FW-AL is proposed for $Ax = 0$ type of constraints (*i.e.,* splitting), but it can be applied to $Ax = b$ case by using a simple product space technique. The analysis of FW-AL relies on the error bounds (see Theorem 1 in (Gidel et al., 2018) for the conditions, and (Bolte et al., 2017) for more details about error bounds). Their dual step-size σ_{k+1} depends on the error bound constant α, as $\sigma_{k+1} = \frac{2\sigma_0}{k+2}$ with $\sigma_0 \leq \min\{\frac{2}{\lambda}, \frac{\alpha^2}{2\delta}\}$. Hence, σ_0 is a tuning parameter, and the method has guaranteed convergence only if it is chosen small enough. However, α is typically not only not known, and it can be also arbitrarily small.

Liu et al. (2018) introduce an inexact augmented Lagrangian method (IAL), where the Lagrangian subproblems are approximately solved by CGM up to a prescribed accuracy $\varepsilon_k = \varepsilon_0/k$ for some $\varepsilon_0 > 0$ to be tuned. This results in a double-loop algorithm, where each outer iteration runs multiple CGM iterations until the following condition is satisfied:

$$\max_{x \in \mathcal{X}} \left\langle \nabla f(x_{k+1}) + A^\top y_k + \lambda A^\top (Ax_{k+1} - b),\, x \right\rangle \leq \varepsilon_k.$$

Then, the algorithm takes a dual gradient ascent step.

IAL provably generates an ϵ-solution after $\mathcal{O}(1/\epsilon^2)$ outer iterations, by choosing the penalty parameter λ appropriately (proportional to $1/\sqrt{\epsilon}$). This method, however, requires multiple queries of lmo at each iteration. Since the number of lmo calls is bounded by $\lceil 6L_f D_{\mathcal{X}}^2/\varepsilon_k \rceil - 2$ (see Theorem 2.2 in (Liu et al., 2018)), the overall lmo complexity of this method is $\mathcal{O}(1/\epsilon^4)$. Note that this is much worse than $\mathcal{O}(1/\epsilon^2)$ calls required by our method.

Yurtsever et al. (2018) present a CG-type method (HCGM) for (P). This method relies on the quadratic penalty approach to handle affine constraints. HCGM guarantees $\mathcal{O}(1/\sqrt{k})$ convergence rate both in the objective residual and the feasibility gap, similar to CGAL. As explained in Section 2, however, penalty methods typically exhibit their proven worst case guarantees in practice. We can indeed observe that the empirical rate of HCGM is $\mathcal{O}(1/\sqrt{k})$ in our numerical experiments (in Section 5), as well as in the experiments in (Yurtsever et al., 2018).

5. Numerical Experiments

This section presents the numerical evidence to demonstrate the empirical superiority of CGAL, based on the max-cut, clustering ,and generalized eigenvector problems.

We compared CGAL against UPD and HCGM from Section 4. This choice is based on the practicality of the algorithms: FW-AL and IAL have 2 tuning parameters each,

and it is very difficult to tune these methods for medium or large scale problems. On the other hand, CGAL, HCGM, and UPD have a single parameter to tune (penalty parameter λ_0 for CGAL and HCGM, and accuracy parameter ϵ for UPD). We tune all these parameters by bisection (with factor 10), until the method (with the chosen parameter) outperforms itself with 10th and $1/10$th of the parameter. Although CGAL with (decr.) performed better than HCGM in all of our experiments, CGAL with (const.) uniformly outperformed (decr.) and HCGM. Hence in this section, we focus on CGAL with (const.).

Note that the computational cost of all algorithms is dominated by lmo. Hence, we plot some of the results with respect to the number of lmo calls. Arguably, this roughly represents the computation time.

5.1. Max-cut

Maximum cut is an NP-Hard combinatorial problem from computer science. Denoting the symmetric $n \times n$ graph Laplacian matrix of a graph by c, this problem can be relaxed as (Goemans and Williamson, 1995):

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \tfrac{1}{4}\operatorname{tr}(cx) \\ \text{subject to} \quad & x_{ii} = 1 \ \text{ for } \ i = 1, 2, \dots, n \\ & \operatorname{tr}(x) = n, \ \ x \in \mathbb{S}^n_+. \end{aligned}$$

Tuning all methods from Section 4 requires substantial computational effort, especially because some of these methods have multiple tuning parameters. To this end, we first consider a small scale max-cut instance where we compare all of these methods (which applies to this problem) and CGAL. In this setup, we use the GD97_b dataset[1], which corresponds to a 47×47 dimensional problem.

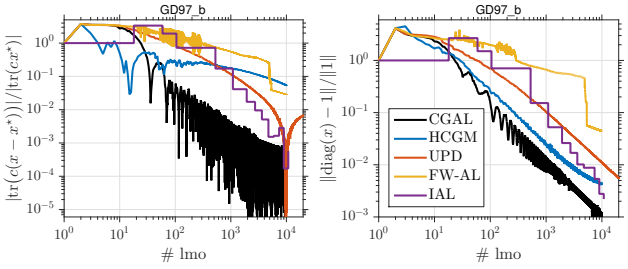

Figure 1. Empirical performance of various methods for solving max-cut problem.

In Figure 1, we present the performance of each method with the best parameter choice obtained after an extensive search. We also provide the performance of each algorithm at all trials in the supplements, and some other variants of these methods as well.

[1]V. Batagelj and A. Mrvar. Pajek datasets, http://vlado.fmf.uni-lj.si/pub/networks/data/

Next, we consider a medium scale experiment, where we compare CGAL, HCGM, and UPD for max-cut with G1 (800×800) and G40 (2000×2000) datasets[2]. We compile the results of these tests in Figure 2. Observe that HCGM converges with $\mathcal{O}(1/\sqrt{k})$ (which is the worst case bound) while CGAL achieves a faster rate.

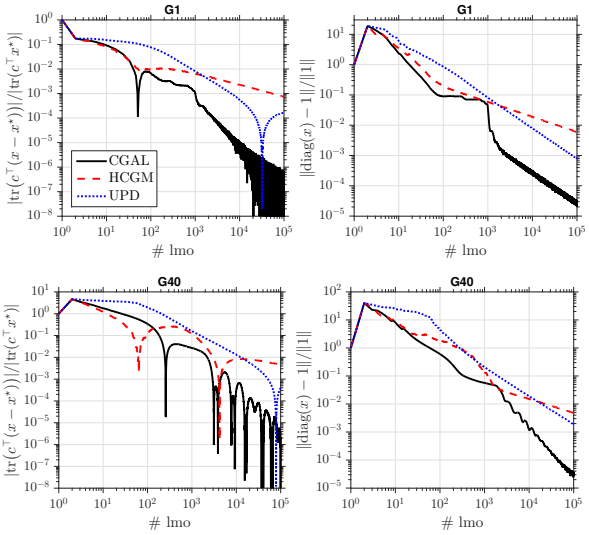

Figure 2. Empirical comparison of CGAL, HCGM and UPD with max-cut problem setup.

5.2. k-means Clustering

Consider the SDP formulation of model-free k-means clustering problem by (Peng and Wei, 2007):

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \operatorname{tr}(cx) \\ \text{subject to} \quad & x1_n = 1_n, \ \ x \geq 0, \ \ x \in \mathbb{S}^n_+ \ \ \& \ \ \operatorname{tr}(x) = \alpha. \end{aligned}$$

where α is the number of clusters and c is the $n \times n$ Euclidean distance matrix. We denote the vector of ones by 1_n, hence $x1_n = 1_n$ and $x \geq 0$ together implies that each row of x is on the unit simplex. The same applies to the columns of x due to symmetry. This problem is an instance of (P), where $f(x) = \operatorname{tr}(cx)$, $\mathcal{X} = \{x : x \in \mathbb{S}^n_+, \operatorname{tr}(x) = \alpha\}$, $A : \mathbb{S}^n_+ \to \mathbb{R}^n \times \mathbb{R}^{n \times n}$ maps $x \to (x1_n, x)$, and $\mathcal{K} = \{1_n\} \times \mathbb{R}^{n \times n}_+$.

We use the same setup as in (Yurtsever et al., 2018), which is designed and published online by Mixon et al. (2017). This setup contains a 1000×1000 dimensional dataset generated by sampling and preprocessing the MNIST dataset[3] using a one-layer neural network. Further details on this setup and the dataset can be found in (Mixon et al., 2017).

[2]Y. Ye. Gset random graphs. https://www.cise.ufl.edu/research/sparse/matrices/gset/

[3]Y. LeCun and C. Cortes. MNIST handwritten digit database, http://yann.lecun.com/exdb/mnist/

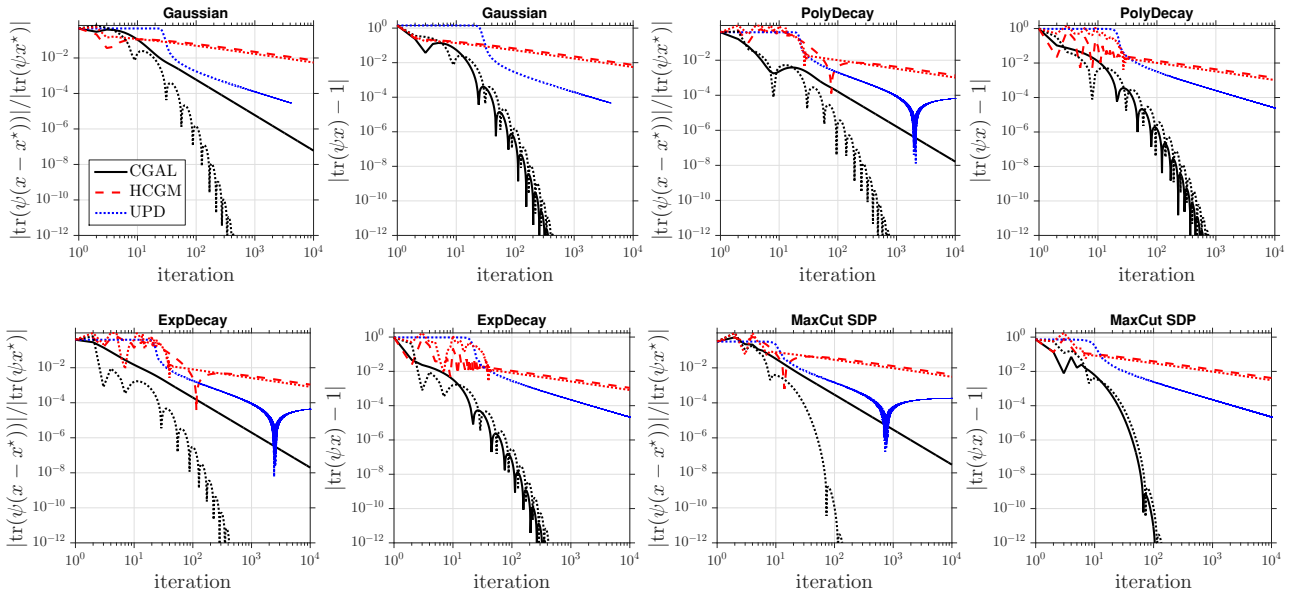

Figure 3. Empirical comparison of CGAL, HCGM and UPD for solving generalized eigenvector problem with 4 different synthetic setups. Dotted lines present objective residual and feasibility gap of the atoms chosen by linear minimization oracle (s_k).

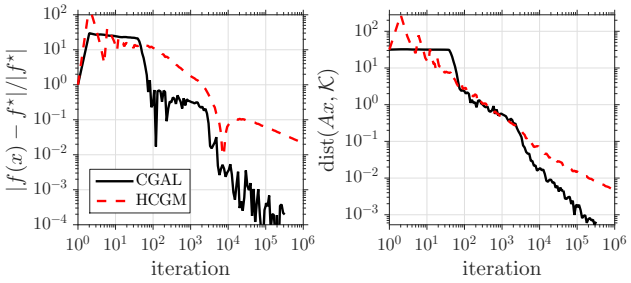

Figure 4. Objective residual and feasibility gap for k-means clustering with preprocessed MNIST dataset.

In Figure 4, we observe once again that CGAL outperforms HCGM, achieving $\mathcal{O}(1/k)$ empirical convergence rate. In this problem instance, we failed to tune UPD, even with the knowledge of $f^\star$. After extensive analysis and tests, we concluded that UPD has an implicit tuning parameter. It is possible to choose different accuracy terms for objective and feasibility in UPD, as also noted by the authors, simply by scaling the objective function with a constant. The performance of UPD heavily depends on this scaling in addition to tuning accuracy parameter, hence we omit UPD.

5.3. Generalized Eigenvector Problem

Consider the SDP relaxation of the generalized eigenvector problem from Boumal et al. (2018):

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \operatorname{tr}(\phi x) \\ \text{subject to} \quad & \operatorname{tr}(x) \le \alpha, \;\; X \in \mathbb{S}^n_+ \;\; \& \;\; \operatorname{tr}(\psi x) = 1 \end{aligned}$$

where ϕ and ψ are symmetric matrices of size $n \times n$ and $\alpha > 0$ is a model parameter. In this problem, we use some synthetic setups, where we generate ψ with *iid* Gaussian entries, and we consider 4 different cases for ϕ:

- `Gaussian` - ϕ generated by taking symmetric part of $10^3 \times 10^3$ *iid* Gaussian matrix
- `PolyDecay` - ϕ generated by randomly rotating $\operatorname{diag}(1^{-i}, 2^{-i}, \ldots, 1000^{-i})$ $(i = 1)$
- `ExpDecay` - ϕ generated by randomly rotating $\operatorname{diag}(10^{-i}, 10^{-2i}, \ldots, 10^{-1000i})$ $(i = 0.025)$
- `MaxCut SDP` - ϕ is a solution of the max-cut SDP with G40 dataset (2000×2000)

This problem highlights an important observation that partially explains the reason why CGAL outperforms the base method HCGM. Remark that this problem has a rank-1 solution, and if we set α correctly, this solution becomes an extreme point of the domain. In this scenario, if the problem is well-conditioned, we might expect the lmo to pick this solution (or some close points). For the sake of the better adaptation to the problem geometry, CGAL updates the dual variable (which corresponds to the center point of a quadratic penalty). In Figure 3, we provide empirical evidence of this adaptation: Dotted lines correspond to extreme points chosen by lmo. Unsurprisingly, these points (s_k) quickly converge (with linear rates) to a solution for CGAL, while we do not observe the same behavior for HCGM or UPD (we omit lmo outputs of UPD in figure which do not converge).

Acknowledgements

This work was supported by the Swiss National Science Foundation (SNSF) under grant number $200021\_178865/1$. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no 725594 - time-data).

References

D. P. Bertsekas. On penalty and multiplier methods for constrained minimization. *SIAM J. Control Optim.*, 14 (2):216–235, 1976.

J. Bolte, T. P. Nguyen, J. Peypouquet, and B. W. Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Math. Program.*, 165(2): 471–507, 2017.

N. Boumal, V. Voroninski, and A. Bandeira. Deterministic guarantees for Burer–Monteiro factorizations of smooth semidefinite programs. arXiv:1804.02008v1, 2018.

E. J. Candès, T. Strohmer, and V. Voroninski. PhaseLift: Exact and stable signal recovery from magnitude measurements via convex programming. *Communications on Pure and Applied Math.*, 66(8):1241–1274, 2013.

C. Dünner, S. Forte, M. Takác, and M. Jaggi. Primal–dual rates and certificates. In *Proc.* 33*rd Int. Conf. Machine Learning*, 2016.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

G. Gidel, F. Pedregosa, and S. Lacoste-Julien. Frank-Wolfe splitting via augmented Lagrangian method. In *Proc.* 21*st Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2018.

M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 43(6):1115–1145, 1995.

E. Hazan and S. Kale. Projection–free online learning. In *Proceedings of the* 29*th International Conference on Machine Learning*, 2012.

X. Huang, I. E.-H. Yen, R. Zhang, Q. Huang, P. Ravikumar, and I. S. Dhillon. Greedy direction method of multiplier for MAP inference of large output domain. In *Proc.* 20*th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2017.

M. Jaggi. Revisiting Frank–Wolfe: Projection–free sparse convex optimization. In *Proc.* 30*th Int. Conf. Machine Learning*, 2013.

N. Komodakis and J.-C. Pesquet. Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *IEEE Signal Process. Mag.*, 32(6):31–54, 2015.

G. Lan. The complexity of large–scale convex programming under a linear optimization oracle. arXiv:1309.5550v2, 2014.

G. Lan and Y. Zhou. Conditional gradient sliding for convex optimization. *SIAM J. Optim.*, 26(2):1379–1409, 2016.

Y.-F. Liu, X. Liu, and S. Ma. On the non-ergodic convergence rate of an inexact augmented Lagrangian framework for composite convex programming. *to appear in Mathematics of Operations Research*, 2018.

D. G. Mixon, S. Villar, and R. Ward. Clustering subgaussian mixtures by semidefinite programming. *Information and Inference: A Journal of the IMA*, 6(4):389–415, 2017.

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1987.

Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103:127–152, 2005a.

Y. Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM J. Optim.*, 16(1):235–249, 2005b.

Y. Nesterov. Universal gradient methods for convex optimization problems. *Math. Program.*, 152(1-2):381–404, 2015.

J. Peng and Y. Wei. Approximating K–means–type clustering via semidefinite programming. *SIAM J. Optim.*, 18 (1):186–205, 2007.

E. K. Ryu and S. Boyd. Primer on monotone operator methods. *Appl. Comput. Math.*, 15(1):3–43, 2016.

M. Sion. On general minimax theorems. *Pacific J. Math.*, 8 (1):171–176, 1958.

Q. Tran-Dinh, O. Fercoq, and V. Cevher. A smooth primal-dual optimization framework for nonsmooth composite convex minimization. *SIAM J. Optim.*, 28(1):96–134, 2018.

S. J. Wright. *Primal–Dual Interior–Point Methods*. SIAM, Philadelphia, USA, 1997.

I. E.-H. Yen, K. Huang, R. Zhong, P. Ravikumar, and I. S. Dhillon. Dual decomposed learning with factorwise oracle for structural svm with large output domain. In *Advances in Neural Information Processing Systems 29*, 2016a.

I. E.-H. Yen, X. Lin, J. Zhang, P. Ravikumar, and I. S. Dhillon. A convex atomic–norm approach to multiple sequence alignment and motif discovery. In *Proceedings of the* 33*rd International Conference on Machine Learning*, 2016b.

A. Yurtsever, Q. Tran-Dinh, and V. Cevher. A universal primal-dual convex optimization framework. In *Advances in Neural Information Processing Systems 28*, 2015.

A. Yurtsever, M. Udell, J. Tropp, and V. Cevher. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *Proc. 20th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2017.

A. Yurtsever, O. Fercoq, F. Locatello, and V. Cevher. A conditional gradient framework for composite convex minimization with applications to semidefinite programming. In *Proc.* 35*th Int. Conf. Machine Learning*, 2018.