Noname manuscript No.
(will be inserted by the editor)

Accelerated gradient sliding for structured convex optimization

Guanghui Lan · Yuyuan Ouyang

the date of receipt and acceptance should be inserted later

Abstract Our main goal in this paper is to show that one can skip gradient computations for gradient descent type methods applied to certain structured convex programming (CP) problems. To this end, we first present an accelerated gradient sliding (AGS) method for minimizing the summation of two smooth convex functions with different Lipschitz constants. We show that the AGS method can skip the gradient computation for one of these smooth components without slowing down the overall optimal rate of convergence. This result is much sharper than the classic black-box CP complexity results especially when the difference between the two Lipschitz constants associated with these components is large. We then consider an important class of bilinear saddle point problem whose objective function is given by the summation of a smooth component and a nonsmooth one with a bilinear saddle point structure. Using the aforementioned AGS method for smooth composite optimization and Nesterov's smoothing technique, we show that one only needs $\mathcal{O}(1/\sqrt{\varepsilon})$ gradient computations for the smooth component while still preserving the optimal $\mathcal{O}(1/\varepsilon)$ overall iteration complexity for solving these saddle point problems. We demonstrate that even more significant savings on gradient computations can be obtained for strongly convex smooth and bilinear saddle point problems.

Keywords convex programming, accelerated gradient sliding, structure, complexity, Nesterov's method

Mathematics Subject Classification (2000) 90C25 · 90C06 · 49M37

1 Introduction

In this paper, we show that one can skip gradient computations without slowing down the convergence of gradient descent type methods for solving certain structured convex programming (CP) problems. To motivate our study, let us first consider the following classic bilinear saddle point problem (SPP):

$$\psi^* := \min_{x\in X}\left\{\psi(x) := f(x) + \max_{y\in Y}\langle Kx, y\rangle - J(y)\right\}. \tag{1.1}$$

The authors are partially supported by National Science Foundation grants 1319050, 1637473 and 1637474, and Office of Naval Research grant N00014-16-1-2802.

Guanghui Lan
H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology
E-mail: george.lan@isye.gatech.edu

Yuyuan Ouyang
Department of Mathematical Sciences, Clemson University
E-mail: yuyuano@clemson.edu

Here, $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ are closed convex sets, $K : \mathbb{R}^n \to \mathbb{R}^m$ is a linear operator, J is a relatively simple convex function, and $f : X \to \mathbb{R}$ is a continuously differentiable convex function satisfying

$$0 \le f(x) - l_f(u, x) \le \frac{L}{2}\|x - u\|^2, \ \forall x, u \in X, \tag{1.2}$$

for some $L > 0$, where $l_f(u, x) := f(u) + \langle \nabla f(u), x - u\rangle$ denotes the first-order Taylor expansion of f at u. Since ψ is a nonsmooth convex function, traditional nonsmooth optimization methods, e.g., the subgradient method, would require $\mathcal{O}(1/\varepsilon^2)$ iterations to find an ε-solution of (1.1), i.e., a point $\bar{x} \in X$ s.t. $\psi(\bar{x}) - \psi^* \le \varepsilon$. In a landmarking work [28], Nesterov suggests to approximate ψ by a smooth convex function

$$\psi_\rho^* := \min_{x \in X} \{\psi_\rho(x) := f(x) + h_\rho(x)\}, \tag{1.3}$$

with

$$h_\rho(x) := \max_{y \in Y} \langle Kx, y\rangle - J(y) - \rho W(y_0, y) \tag{1.4}$$

for some $\rho > 0$, where $y_0 \in Y$ and $W(y_0, \cdot)$ is a strongly convex function. By properly choosing ρ and applying the optimal gradient method to (1.3), he shows that one can compute an ε-solution of (1.1) in at most

$$\mathcal{O}\left(\sqrt{\frac{L}{\varepsilon}} + \frac{\|K\|}{\varepsilon}\right) \tag{1.5}$$

iterations. Following [28], much research effort has been devoted to the development of first-order methods utilizing the saddle-point structure of (1.1) (see, e.g., the smoothing technique [28,25,1,21,8,17,31,2,19], the mirror-prox methods [23,6,14,18], the primal-dual type methods [5,31,9,32,7,16] and their equivalent form as the alternating direction method of multipliers [22,12,13,29,30,15]). Some of these methods (e.g., [7,6,30,16, 19]) can achieve exactly the same complexity bound as in (1.5).

One problem associated with Nesterov's smoothing scheme and the related methods mentioned above is that each iteration of these methods require both the computation of ∇f and the evaluation of the linear operators (K and K^T). As a result, the total number of gradient and linear operator evaluations will both be bounded by $\mathcal{O}(1/\varepsilon)$. However, in many applications the computation of ∇f is often much more expensive than the evaluation of the linear operators K and K^T. This happens, for example, when the linear operator K is sparse (e.g., total variation, overlapped group lasso and graph regularization), while f involves a more expensive data-fitting term (see Section 4 and [20] for some other examples). In [20], Lan considered some similar situation and proposed a gradient sliding (GS) algorithm to minimize a class of composite problems whose objective function is given by the summation of a general smooth and nonsmooth component. He shows that one can skip the computation of the gradient for the smooth component from time to time, while still maintaining the $\mathcal{O}(1/\varepsilon^2)$ iteration complexity bound. More specifically, by applying the GS method in [20] to problem (1.1), we can show that the number of gradient evaluations of ∇f will be bounded by

$$\mathcal{O}\left(\sqrt{\frac{L}{\varepsilon}}\right),$$

which is significantly better than (1.5). Unfortunately, the total number of evaluations for the linear operators K and K^T will be bounded by

$$\mathcal{O}\left(\sqrt{\frac{L}{\varepsilon}} + \frac{\|K\|^2}{\varepsilon^2}\right),$$

which is much worse than (1.5). An important yet unresolved research question is whether one can still preserve the optimal $\mathcal{O}(1/\varepsilon)$ complexity bound in (1.5) for solving (1.1) by utilizing only $\mathcal{O}(1/\sqrt{\varepsilon})$ gradient computations of ∇f to find an ε-solution of (1.1). If so, we could be able to keep the total number of iterations relatively small, but significantly reduce the total number of required gradient computations.

In order to address the aforementioned issues associated with existing solution methods for solving (1.1), we pursue in this paper a different approach to exploit the structural information of (1.1). Firstly, instead of concentrating solely on nonsmooth optimization as in [20], we study the following smooth composite optimization problem:

$$\phi^* := \min_{x\in X} \{\phi(x) := f(x) + h(x)\}. \tag{1.6}$$

Here f and h are smooth convex functions satisfying (1.2) and

$$0 \le h(x) - l_h(u,x) \le \frac{M}{2}\|x-u\|^2, \ \forall x,u \in X, \tag{1.7}$$

respectively. It is worth noting that problem (1.6) can be viewed as a special cases of (1.1) or (1.3) (with $J = h^*$ being a strongly convex function, $Y = \mathbb{R}^n$, $K = I$ and $\rho = 0$). Under the assumption that $M \ge L$, we present a novel accelerated gradient sliding (AGS) method which can skip the computation of ∇f from time to time. We show that the total number of required gradient evaluations of ∇f and ∇h, respectively, can be bounded by

$$\mathcal{O}\left(\sqrt{\frac{L}{\varepsilon}}\right) \quad \text{and} \quad \mathcal{O}\left(\sqrt{\frac{M}{\varepsilon}}\right) \tag{1.8}$$

to find an ε-solution of (1.6). Observe that the above complexity bounds are sharper than the complexity bound obtained by Nesterov's optimal method for smooth convex optimization, which is given by

$$\mathcal{O}\left(\sqrt{\frac{L+M}{\varepsilon}}\right).$$

In particular, for the AGS method, the Lipschitz constant M associated with ∇h does not affect at all the number of gradient evaluations of ∇f. Clearly, the higher ratio of M/L will potentially result in more savings on the gradient computation of ∇f. Moreover, if f is strongly convex with modulus μ, then the above two complexity bounds in (1.8) can be significantly reduced to

$$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}\right) \quad \text{and} \quad \mathcal{O}\left(\sqrt{\frac{M}{\mu}}\log\frac{1}{\varepsilon}\right), \tag{1.9}$$

respectively, which also improves Nesterov's optimal method applied to (1.6) in terms of the number gradient evaluations of ∇f. Observe that in the classic black-box setting [24,27] the complexity bounds in terms of gradient evaluations of ∇f and ∇h are intertwined, and a larger Lipschitz constant M will result in more gradient evaluations of ∇f, even though there is no explicit relationship between ∇f and M. In our development, we break down the black-box assumption by assuming that we have separate access to ∇f and ∇h rather than $\nabla \phi$ as a whole. To the best of our knowledge, these types of separate complexity bounds as in (1.8) and (1.9) have never been obtained before for smooth convex optimization.

Secondly, we apply the above AGS method to the smooth approximation problem (1.3) in order to solve the aforementioned bilinear SPP in (1.1). By choosing the smoothing parameter properly, we show that the total number of gradient evaluations of ∇f and operator evaluations of K (and K^T) for finding an ε-solution of (1.1) can be bounded by

$$\mathcal{O}\left(\sqrt{\frac{L}{\varepsilon}}\right) \quad \text{and} \quad \mathcal{O}\left(\frac{\|K\|}{\varepsilon}\right),$$

respectively. In comparison with Nesterov's original smoothing scheme and other existing methods for solving (1.1), our method can provide significant savings on the number of gradient computations of ∇f without increasing the complexity bound on the number of operator evaluations of K and K^T. In comparison with the GS method in [20], our method can reduce the number of operator evaluations of K and K^T from $\mathcal{O}(1/\varepsilon^2)$ to $\mathcal{O}(1/\varepsilon)$. Moreover, if f is strongly convex with modulus μ, the above two bounds will be significantly reduced to

$$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}\right) \quad \text{and} \quad \mathcal{O}\left(\frac{\|K\|}{\sqrt{\varepsilon}}\right),$$

respectively. To the best of our knowledge, this is the first time that these tight complexity bounds were obtained for solving the classic bilinear saddle point problem (1.1).

It should be noted that, even though the idea of skipping the computation of ∇f is similar to [20], the AGS method presented in this paper significantly differs from the GS method in [20]. In particular, each iteration of GS method consists of one accelerated gradient iteration together with a bounded number of subgradient iterations. On the other hand, each iteration of the AGS method is composed of an accelerated gradient iteration nested with a few other accelerated gradient iterations to solve a different subproblem. The development of the AGS method seems to be more technical than GS and its convergence analysis is also highly nontrivial.

This paper is organized as follows. We first present the AGS method and discuss its convergence properties for minimizing the summation of two smooth convex functions (1.6) in Section 2. Utilizing this new algorithm and its associated convergence results, we study the properties of the AGS method for minimizing the bilinear saddle point problem (1.1) in Section 3. We then demonstrate the effectiveness of the AGS method through out preliminary numerical experiments for solving certain portfolio optimization and image reconstruction problems in Section 4. Some brief concluding remarks are made in Section 5.

Notation, assumption and terminology

We use $\|\cdot\|$, $\|\cdot\|_*$, and $\langle\cdot,\cdot\rangle$ to denote an arbitrary norm, the associated conjugate and the inner product in Euclidean space $\mathbb{R}^n$, respectively. For any set X, we say that $V(\cdot,\cdot)$ is a prox-function associated with $X \subseteq \mathbb{R}^n$ modulus ν if there exists a strongly convex function $\pi(\cdot)$ with strong convexity parameter ν such that

$$V(x,u) = \pi(u) - \pi(x) - \langle \nabla\pi(x), u - x\rangle, \ \forall x, u \in X. \tag{1.10}$$

The above prox-function is also known as the Bregman divergence [3] (see also [28,23]), which generalizes the Euclidean distance $\|x-u\|_2^2/2$. It can be easily seen from (1.10) and the strong convexity of π that

$$V(x,u) \geq \frac{\nu}{2}\|x-u\|^2 \quad \forall x, y \in X. \tag{1.11}$$

Moreover, we say that the prox-function grows quadratically if there exists a constant C such that $V(x,u) \leq C\|x-u\|^2/2$. Without loss of generality, we assume that $C = 1$ whenever this happens, i.e.,

$$V(x,u) \leq \frac{1}{2}\|x-u\|^2. \tag{1.12}$$

In this paper, we associate sets $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ with prox-functions $V(\cdot,\cdot)$ and $W(\cdot,\cdot)$ with moduli ν and ω w.r.t. their respective norms in $\mathbb{R}^n$ and $\mathbb{R}^m$.

For any real number r, $\lceil r \rceil$ and $\lfloor r \rfloor$ denote the nearest integer to r from above and below, respectively. We denote the set of nonnegative and positive real numbers by $\mathbb{R}_+$ and $\mathbb{R}_{++}$, respectively.

2 Accelerated gradient sliding for composite smooth optimization

In this section, we present an accelerated gradient sliding (AGS) algorithm for solving the smooth composite optimization problem in (1.6) and discuss its convergence properties. Our main objective is to show that the AGS algorithm can skip the evaluation of ∇f from time to time and achieve better complexity bounds in terms of gradient computations than the classical optimal first-order methods applied to (1.6) (e.g., Nesterov's method in [26]). Without loss of generality, throughout this section we assume that $M \geq L$ in (1.2) and (1.7).

The AGS method evolves from the gradient sliding (GS) algorithm in [20], which was designed to solve a class of composite convex optimization problems with the objective function given by the summation of a smooth and nonsmooth component. The basic idea of the GS method is to keep the nonsmooth term inside the projection (or proximal mapping) in the accelerated gradient method and then to apply a few subgradient descent iterations to solve the projection subproblem. Inspired by [20], we suggest to keep the smooth term h that has a larger Lipschitz constant in the proximal mapping in the accelerated gradient method, and then to apply a few accelerated gradient iterations to solve this smooth subproblem. As a consequence, the proposed AGS method

involves two nested loops (i.e., outer and inner iterations), each of which consists of a set of modified accelerated gradient descent iterations (see Algorithm 1). At the k-th outer iteration, we first build a linear approximation $g_k(u) = l_f(\underline{x}_k, u)$ of f at the search point $\underline{x}_k \in X$ and then call the ProxAG procedure in (2.4) to compute a new pair of search points $(x_k, \tilde{x}_k) \in X \times X$. The ProxAG procedure can be viewed as a subroutine to compute a pair of approximate solutions to

$$\min_{u \in X} g_k(u) + h(u) + \beta V(x_{k-1}, u), \tag{2.1}$$

where $g_k(\cdot)$ is defined in (2.3), and x_{k-1} is called the prox-center at the k-th outer iteration. It is worth mentioning that there are two essential differences associated with the steps (2.2)-(2.6) from the standard Nesterov's accelerated gradient iterations. Firstly, we use two different search points, i.e., x_k and $\overline{x}_k$, respectively, to update $\underline{x}_k$ to compute the linear approximation and $\overline{x}_k$ to compute the output solution in (2.5). Secondly, we employ two parameters, i.e., γ_k and λ_k, to update $\underline{x}_k$ and $\overline{x}_k$, respectively, rather than just one single parameter.

The ProxAG procedure in Algorithm 1 performs T_k inner accelerated gradient iterations to solve (2.1) with certain properly chosen starting points $\tilde{u}_0$ and u_0. It should be noted, however, that the accelerated gradient iterations in (2.6)-(2.8) also differ from the standard Nesterov's accelerated gradient iterations in the sense that the definition of the search point $\underline{u}_t$ involves a fixed search point $\overline{x}$. Since each inner iteration of the ProxAG procedure requires one evaluation of ∇h and no evaluation of ∇f, the number of gradient evaluations of ∇h will be greater than that of ∇f as long as $T_k > 1$. On the other hand, if $\lambda_k \equiv \gamma_k$ and $T_k \equiv 1$ in the AGS method, and $\alpha_t \equiv 1$, and $p_t \equiv q_t \equiv 0$ in the ProxAG procedure, then (2.4) becomes

$$x_k = \tilde{x}_k = \operatorname*{argmin}_{u \in X} g_k(u) + l_h(\underline{x}_k, u) + \beta_k V(x_{k-1}, u).$$

In this case, the AGS method reduces to a variant of Nesterov's optimal gradient method (see, e.g., [27,31]).

Algorithm 1 Accelerated gradient sliding (AGS) algorithm for solving (1.6)

Choose $x_0 \in X$. Set $\overline{x}_0 = x_0$.
for $k = 1, \ldots, N$ **do**

$$\underline{x}_k = (1-\gamma_k)\overline{x}_{k-1} + \gamma_k x_{k-1}, \tag{2.2}$$
$$g_k(\cdot) = l_f(\underline{x}_k, \cdot), \tag{2.3}$$
$$(x_k, \tilde{x}_k) = ProxAG(g_k, \overline{x}_{k-1}, x_{k-1}, \lambda_k, \beta_k, T_k) \tag{2.4}$$
$$\overline{x}_k = (1-\lambda_k)\overline{x}_{k-1} + \lambda_k \tilde{x}_k. \tag{2.5}$$

end for
Output $\overline{x}_N$.

procedure $(x^+, \tilde{x}^+) = ProxAG(g, \overline{x}, x, \lambda, \beta, \gamma, T)$
Set $\tilde{u}_0 = \overline{x}$ and $u_0 = x$.
for $t = 1, \ldots, T$ **do**

$$\underline{u}_t = (1-\lambda)\overline{x} + \lambda(1-\alpha_t)\tilde{u}_{t-1} + \lambda\alpha_t u_{t-1}, \tag{2.6}$$
$$u_t = \operatorname*{argmin}_{u \in X} g(u) + l_h(\underline{u}_t, u) + \beta V(x, u) + (\beta p_t + q_t) V(u_{t-1}, u), \tag{2.7}$$
$$\tilde{u}_t = (1-\alpha_t)\tilde{u}_{t-1} + \alpha_t u_t, \tag{2.8}$$

end for
Output $x^+ = u_T$ and $\tilde{x}^+ = \tilde{u}_T$.
end procedure

Our goal in the remaining part of this section is to establish the convergence of the AGS method and to provide theoretical guidance to specify quite a few parameters, including $\{\gamma_k\}$, $\{\beta_k\}$, $\{T_k\}$, $\{\lambda_k\}$, $\{\alpha_t\}$, $\{p_t\}$, and $\{q_t\}$, used in the generic statement of this algorithm. In particular, we will provide upper bounds on the number of outer and inner iterations, corresponding to the number of gradient evaluations of ∇f and ∇h, respectively, performed by the AGS method to find an ε-solution to (1.6).

We will first study the convergence properties of the ProxAG procedure from which the convergence of the AGS method immediately follows. In our analysis, we measure the quality of the output solution computed at the k-th call to the ProxAG procedure by

$$Q_k(x,u) := g_k(x) - g_k(u) + h(x) - h(u). \tag{2.9}$$

Indeed, if x^* is an optimal solution to (1.6), then $Q_k(x,x^*)$ provides a linear approximation for the functional optimality gap $\phi(x)-\phi(x^*) = f(x)-f(x^*)+h(x)-h(x^*)$ obtained by replacing f with g_k. The following result describes some relationship between $\phi(x)$ and $Q_k(\cdot,\cdot)$.

Lemma 2.1 *For any $u \in X$, we have*

$$\begin{aligned}
&\phi(\overline{x}_k) - \phi(u) \\
\le&(1-\gamma_k)[\phi(\overline{x}_{k-1}) - \phi(u)] + Q_k(\overline{x}_k, u) - (1-\gamma_k)Q_k(\overline{x}_{k-1}, u) \\
&+ \frac{L}{2}\|\overline{x}_k - \underline{x}_k\|^2.
\end{aligned} \tag{2.10}$$

Proof By (1.2), (1.6), (2.3), and the convexity of $f(\cdot)$, we have

$$\begin{aligned}
&\phi(\overline{x}_k) - (1-\gamma_k)\phi(\overline{x}_{k-1}) - \gamma_k\phi(u) \\
\le& g_k(\overline{x}_k) + \frac{L}{2}\|\overline{x}_k - \underline{x}_k\|^2 + h(\overline{x}_k) \\
&- (1-\gamma_k)f(\overline{x}_{k-1}) - (1-\gamma_k)h(\overline{x}_{k-1}) - \gamma_k f(u) - \gamma_k h(u) \\
\le& g_k(\overline{x}_k) + \frac{L}{2}\|\overline{x}_k - \underline{x}_k\|^2 + h(\overline{x}_k) \\
&- (1-\gamma_k)g_k(\overline{x}_{k-1}) - (1-\gamma_k)h(\overline{x}_{k-1}) - \gamma_k g_k(u) - \gamma_k h(u) \\
=& Q_k(\overline{x}_k, u) - (1-\gamma_k)Q_k(\overline{x}_{k-1}, u) + \frac{L}{2}\|\overline{x}_k - \underline{x}_k\|^2.
\end{aligned}$$

□

In order to analyze the ProxAG procedure, we need the following two technical results. The first one below characterizes the solution of optimization problems involving prox-functions. The proof of this result can be found, for example, in Lemma 2 of [10].

Lemma 2.2 *Suppose that a convex set $Z \subseteq \mathbb{R}^n$, a convex function $q: Z \to \mathbb{R}$, points $z, z' \in Z$ and scalars $\mu_1, \mu_2 \in \mathbb{R}_+$ are given. Also let $V(z,u)$ be a prox-function. If*

$$u^* \in \operatorname*{Argmin}_{u \in Z} q(u) + \mu_1 V(z,u) + \mu_2 V(z',u),$$

then for any $u \in Z$, we have

$$\begin{aligned}
&q(u^*) + \mu_1 V(z,u^*) + \mu_2 V(z',u^*) \\
\le& q(u) + \mu_1 V(z,u) + \mu_2 V(z',u) - (\mu_1+\mu_2)V(u^*,u).
\end{aligned}$$

The second technical result slightly generalizes Lemma 3 of [19] to provide a convenient way to study sequences with sublinear rates of convergence.

Lemma 2.3 *Let $c_k \in (0,1]$, $k = 1, 2, \ldots$ and $C_1 > 0$ be given, and define*

$$C_k := (1 - c_k)C_{k-1}, \ k \geq 2.$$

Suppose that $C_k > 0$ for all $k \geq 2$ and that the sequence $\{\delta_k\}_{k\geq 0}$ satisfies

$$\delta_k \leq (1 - c_k)\delta_{k-1} + B_k, \ k = 1, 2, \ldots. \tag{2.11}$$

For any $k \geq 1$, we have

$$\delta_k \leq C_k \left[\frac{1 - c_1}{C_1}\delta_0 + \sum_{i=1}^{k} \frac{B_i}{C_i} \right]. \tag{2.12}$$

In particular, the above inequality becomes equality when the relations in (2.11) are all equality relations.

Proof The result follows from dividing both sides of (2.11) by C_k and then summing up the resulting inequalities or equalities. □

It should be noted that, although (2.11) and (2.12) are stated in the form of inequalities, we can derive some useful formulas by setting them to be equalities. For example, let $\{\alpha_t\}$ be the parameters used in the ProxAG procedure (see (2.6) and (2.8)) and consider the sequence $\{\Lambda_t\}_{t\geq 1}$ defined by

$$\Lambda_t = \begin{cases} 1 & t = 1, \\ (1 - \alpha_t)\Lambda_{t-1} & t > 1. \end{cases} \tag{2.13}$$

By Lemma 2.3 (with $k = t$, $C_k = \Lambda_t$, $c_k = \alpha_t$, $\delta_k \equiv 1$, and $B_k = \alpha_t$), we have

$$1 = \Lambda_t \left[\frac{1 - \alpha_1}{\Lambda_1} + \sum_{i=1}^{t} \frac{\alpha_i}{\Lambda_i} \right] = \Lambda_t(1 - \alpha_1) + \Lambda_t \sum_{i=1}^{t} \frac{\alpha_i}{\Lambda_i}, \tag{2.14}$$

where the last identity follows from the fact that $\Lambda_1 = 1$ in (2.13). Similarly, applying Lemma 2.3 to the recursion $\tilde{u}_t = (1 - \alpha_t)\tilde{u}_{t-1} + \alpha_t u_t$ in (2.8) (with $k = t$, $C_k = \Lambda_t$, $c_k = \alpha_t$, $\delta_k = \tilde{u}_t$, and $B_k = \alpha_t u_t$), we have

$$\tilde{u}_t = \Lambda_t \left[(1 - \alpha_1)\tilde{u}_0 + \sum_{i=1}^{t} \frac{\alpha_i}{\Lambda_i} u_i \right]. \tag{2.15}$$

In view of (2.14) and the fact that $\tilde{u}_0 = \overline{x}$ in the description of the ProxAG procedure, the above relation indicates that $\tilde{u}_t$ is a convex combination of $\overline{x}$ and $\{u_i\}_{i=1}^t$.

With the help of the above two technical results, we are now ready to derive some important convergence properties for the ProxAG procedure in terms of the error measure $Q_k(\cdot, \cdot)$. For the sake of notational convenience, when we work on the k-th call to the ProxAG procedure, we drop the subscript k in (2.9) and just denote

$$Q(x, u) := g(x) - g(u) + h(x) - h(x). \tag{2.16}$$

In a similar vein, we also define

$$\underline{x} := (1 - \gamma)\overline{x} + \gamma x \text{ and } \overline{x}^+ := (1 - \lambda)\overline{x} + \lambda\tilde{x}^+. \tag{2.17}$$

Comparing the above notations with (2.2) and (2.5), we can observe that $\underline{x}$ and $\overline{x}^+$, respectively, represent $\underline{x}_k$ and $\overline{x}_k$ in the k-th call to the ProxAG procedure.

Lemma 2.4 *Consider the k-th call to the ProxAG procedure in Algorithm 1 and let Λ_t and $\overline{x}^+$ be defined in (2.13) and (2.17) respectively. If the parameters satisfy*

$$\lambda \le 1, \Lambda_T(1-\alpha_1) = 1 - \frac{\gamma}{\lambda}, \text{ and } \beta p_t + q_t \ge \frac{\lambda M \alpha_t}{\nu}, \tag{2.18}$$

then

$$Q(\overline{x}^+, u) - (1-\gamma)Q(\overline{x}, u) \le \Lambda_T \sum_{t=1}^{T} \frac{\Upsilon_t(u)}{\Lambda_t}, \ \forall u \in X, \tag{2.19}$$

where

$$\begin{aligned}\Upsilon_t(u) :=& \lambda\beta\alpha_t[V(x,u) - V(x,u_t) + p_t V(u_{t-1},u) - (1+p_t)V(u_t,u)] \\ &+ \lambda\alpha_t q_t[V(u_{t-1},u) - V(u_t,u)].\end{aligned} \tag{2.20}$$

Proof Let us fix any arbitrary $u \in X$ and denote

$$v := (1-\lambda)\overline{x} + \lambda u, \text{ and } \overline{u}_t := (1-\lambda)\overline{x} + \lambda \tilde{u}_t. \tag{2.21}$$

Our proof consists of two major parts. We first prove that

$$Q(\overline{x}^+, u) - (1-\gamma)Q(\overline{x}, u) \le Q(\overline{u}_T, v) - \left(1 - \frac{\lambda}{\gamma}\right) Q(\overline{u}_0, v), \tag{2.22}$$

and then estimate the right-hand-side of (2.22) through the following recurrence property:

$$Q(\overline{u}_t, v) - (1-\alpha_t)Q(\overline{u}_{t-1}, v) \le \Upsilon_t(u). \tag{2.23}$$

The result in (2.19) then follows as an immediate consequence of (2.22) and (2.23). Indeed, by Lemma 2.3 applied to (2.23) (with $k = t$, $C_k = \Lambda_t$, $c_k = \alpha_t$, $\delta_k = Q(\overline{u}_t, v)$, and $B_k = \Upsilon_t(u)$), we have

$$\begin{aligned}Q(\overline{u}_T, v) \le& \Lambda_T \left[\frac{1-\alpha_1}{\Lambda_1} Q(\overline{u}_0, v) - \sum_{t=1}^{T} \frac{\Upsilon_t(u)}{\Lambda_t}\right] \\ =& \left(1 - \frac{\lambda}{\gamma}\right) Q(\overline{u}_0, v) - \Lambda_T \sum_{t=1}^{T} \frac{\Upsilon_t(u)}{\Lambda_t},\end{aligned}$$

where last inequality follows from (2.18) and the fact that $\Lambda_1 = 1$ in (2.13). The above relation together with (2.22) then clearly imply (2.19).

We start with the first part of the proof regarding (2.22). By (2.16) and the linearity of $g(\cdot)$, we have

$$\begin{aligned}&Q(\overline{x}^+, u) - (1-\gamma)Q(\overline{x}, u) \\ =& g(\overline{x}^+ - (1-\gamma)\overline{x} - \gamma u) + h(\overline{x}^+) - (1-\gamma)h(\overline{x}) - \gamma h(u) \\ =& g(\overline{x}^+ - \overline{x} + \gamma(\overline{x} - u)) + h(\overline{x}^+) - h(\overline{x}) + \gamma(h(\overline{x}) - h(u)).\end{aligned} \tag{2.24}$$

Now, noting that by the relation between u and v in (2.21), we have

$$\gamma(\overline{x} - u) = \frac{\gamma}{\lambda}(\lambda\overline{x} - \lambda u) = \frac{\gamma}{\lambda}(\overline{x} - v). \tag{2.25}$$

In addition, by (2.21) and the convexity of $h(\cdot)$, we obtain

$$\frac{\gamma}{\lambda}[h(v) - (1-\lambda)h(\overline{x}) - \lambda h(u)] \le 0,$$

or equivalently,

$$\gamma(h(\overline{x}) - h(u)) \le \frac{\gamma}{\lambda}(h(\overline{x}) - h(v)). \tag{2.26}$$

Applying (2.25) and (2.26) to (2.24), and using the definition of $Q(\cdot,\cdot)$ in (2.16), we obtain

$$Q(\overline{x}^+,u)-(1-\gamma)Q(\overline{x},u)\le Q(\overline{x}^+,v)-\left(1-\frac{\lambda}{\gamma}\right)Q(\overline{x},v).$$

Noting that $\tilde{u}_0=\overline{x}$ and $\tilde{x}=\tilde{u}_T$ in the description of the ProxAG procedure, by (2.17) and (2.21) we have $\overline{x}^+=\overline{u}_T$ and $\overline{u}_0=\overline{x}$. Therefore, the above relation is equivalent to (2.22), and we conclude the first part of the proof.

For the second part of the proof regarding (2.23), first observe that by the definition of $Q(\cdot,\cdot)$ in (2.16), the convexity of $h(\cdot)$, and (1.7),

$$\begin{aligned}
&Q(\overline{u}_t,v)-(1-\alpha_t)Q(\overline{u}_{t-1},v)\\
=&\lambda\alpha_t(g(u_t)-g(u))+h(\overline{u}_t)-(1-\alpha_t)h(\overline{u}_{t-1})-\alpha_t h(v)\\
\le&\lambda\alpha_t(g(u_t)-g(u))+l_h(\underline{u}_t,\overline{u}_t)+\frac{M}{2}\|\overline{u}_t-\underline{u}_t\|^2\\
&-(1-\alpha_t)l_h(\underline{u}_t,\overline{u}_{t-1})-\alpha_t l_h(\underline{u}_t,v)\\
=&\lambda\alpha_t(g(u_t)-g(u))+l_h(\underline{u}_t,\overline{u}_t-(1-\alpha_t)\overline{u}_{t-1}-\alpha_t v)+\frac{M}{2}\|\overline{u}_t-\underline{u}_t\|^2.
\end{aligned}\tag{2.27}$$

Also note that by (2.6), (2.8), and (2.21),

$$\begin{aligned}
&\overline{u}_t-(1-\alpha_t)\overline{u}_{t-1}-\alpha_t v=(\overline{u}_t-\overline{u}_{t-1})+\alpha_t(\overline{u}_{t-1}-v)\\
=&\lambda(\tilde{u}_t-\tilde{u}_{t-1})+\lambda\alpha_t(\tilde{u}_{t-1}-u)=\lambda(\tilde{u}_t-(1-\alpha_t)\tilde{u}_{t-1})-\lambda\alpha_t u\\
=&\lambda\alpha_t(u_t-u).
\end{aligned}$$

By a similar argument as the above, we have

$$\overline{u}_t-\underline{u}_t=\lambda(\tilde{u}_t-(1-\alpha_t)\tilde{u}_{t-1})-\lambda\alpha_t u_{t-1}=\lambda\alpha_t(u_t-u_{t-1}).$$

Using the above two identities in (2.27), we have

$$\begin{aligned}
&Q(\overline{u}_t,v)-(1-\alpha_t)Q(\overline{u}_{t-1},v)\\
\le&\lambda\alpha_t\left[g(u_t)-g(u)+l_h(\underline{u}_t,u_t)-l_h(\underline{u}_t,u)+\frac{M\lambda\alpha_t}{2}\|u_t-u_{t-1}\|^2\right].
\end{aligned}$$

Moreover, it follows from Lemma 2.2 applied to (2.7) that

$$\begin{aligned}
&g(u_t)-g(u)+l_h(\underline{u}_t,u_t)-l_h(\underline{u}_t,u)\\
\le&\beta(V(x,u)-V(u_t,u)-V(x,u_t))\\
&+(\beta p_t+q_t)(V(u_{t-1},u)-V(u_t,u)-V(u_{t-1},u_t)).
\end{aligned}$$

Also by (1.11) and (2.18), we have

$$\frac{M\lambda\alpha_t}{2}\|u_t-u_{t-1}\|^2\le\frac{M\lambda\alpha_t}{2\nu}V(u_{t-1},u_t)\le(\beta p_t+q_t)V(u_{t-1},u_t).$$

Combining the above three relations, we conclude (2.23). □

In the following proposition, we provide certain sufficient conditions under which the the right-hand-side of (2.19) can be properly bounded. As a consequence, we obtain a recurrence relation for the ProxAG procedure in terms of $Q(\overline{x}_k,u)$.

Proposition 2.1 *Consider the k-th call to the ProxAG procedure. If (2.18) holds,*

$$\frac{\alpha_t q_t}{\Lambda_t} = \frac{\alpha_{t+1} q_{t+1}}{\Lambda_{t+1}} \quad \text{and} \quad \frac{\alpha_t(1+p_t)}{\Lambda_t} = \frac{\alpha_{t+1} p_{t+1}}{\Lambda_{t+1}} \tag{2.28}$$

for any $1 \le t \le T-1$, then we have

$$\begin{aligned} & Q(\overline{x}^+, u) - (1-\gamma) Q(\overline{x}, u) \\ \le & \lambda \alpha_T [\beta(1+p_T) + q_T] \left[V(x,u) - V(x^+, u) \right] - \frac{\nu\beta}{2\gamma} \|\overline{x}^+ - \underline{x}\|^2, \end{aligned} \tag{2.29}$$

where $\overline{x}^+$ and $\underline{x}$ are defined in (2.17).

Proof To prove the proposition it suffices to estimate the right-hand-side of (2.19). We make three observations regarding the terms in (2.19) and (2.20). First, by (2.14),

$$\lambda\beta\Lambda_T \sum_{t=1}^{T} \frac{\alpha_t}{\Lambda_t} V(x,u) = \lambda\beta(1 - \Lambda_T(1-\alpha_1)) V(x,u).$$

Second, by (1.11), (2.14), (2.15), (2.18), and the fact that $\tilde{u}_0 = \overline{x}$ and $\tilde{x}^+ = \tilde{u}_T$ in the ProxAG procedure, we have

$$\begin{aligned} \lambda\beta\Lambda_T \sum_{t=1}^{T} \frac{\alpha_t}{\Lambda_t} V(x, u_t) \ge & \frac{\nu\gamma\beta}{2} \cdot \frac{\Lambda_T}{(1 - \Lambda_T(1-\alpha_1))} \sum_{t=1}^{T} \frac{\alpha_t}{\Lambda_t} \|x - u_t\|^2 \\ \ge & \frac{\nu\gamma\beta}{2} \left\| x - \frac{\Lambda_T}{1 - \Lambda_T(1-\alpha_1)} \sum_{i=1}^{T} \frac{\alpha_t}{\Lambda_t} u_t \right\|^2 \\ = & \frac{\nu\gamma\beta}{2} \left\| x - \frac{\tilde{u}_T - \Lambda_T(1-\alpha_1)\tilde{u}_0}{1 - \Lambda_T(1-\alpha_1)} \right\|^2 \\ = & \frac{\nu\gamma\beta}{2} \left\| x - \frac{\lambda}{\gamma}\tilde{u}_T - \left(1 - \frac{\lambda}{\gamma}\right)\tilde{u}_0 \right\|^2 \\ = & \frac{\nu\beta}{2\gamma} \left\| \gamma x - \lambda \tilde{x}^+ - (\gamma - \lambda)\overline{x} \right\|^2 \\ = & \frac{\nu\beta}{2\gamma} \|\underline{x} - \overline{x}^+\|^2, \end{aligned}$$

where the last equality follows from (2.17). Third, by (2.28), the fact that $\Lambda_1 = 1$ in (2.13), and the relations that $u_0 = x$ and $u_T = x^+$ in the ProxAG procedure, we have

$$\begin{aligned} & \lambda\beta\Lambda_T \sum_{t=1}^{T} \frac{\alpha_t}{\Lambda_t} [p_t V(u_{t-1}, u) - (1+p_t) V(u_t, u)] \\ & + \lambda\Lambda_T \sum_{t=1}^{T} \frac{\alpha_t q_t}{\Lambda_t} [V(u_{t-1}, u) - V(u_t, u)] \\ = & \lambda\beta\Lambda_T \left[\alpha_1 p_1 V(u_0, u) - \sum_{i=1}^{T-1} \left(\frac{\alpha_t(1+p_t)}{\Lambda_t} - \frac{\alpha_{t+1} p_{t+1}}{\Lambda_{t+1}} \right) V(u_t, u) \right. \\ & \left. - \frac{\alpha_T(1+p_T)}{\Lambda_T} V(u_T, u) \right] + \lambda\alpha_T q_T [V(u_0, u) - V(u_T, u)] \\ = & \lambda\beta \left[\Lambda_T \alpha_1 p_1 V(u_0, u) - \alpha_T(1+p_T) V(u_T, u) \right] + \lambda\alpha_T q_T [V(u_0, u) - V(u_T, u)] \\ = & \lambda\beta \left[\Lambda_T \alpha_1 p_1 V(x, u) - \alpha_T(1+p_T) V(x^+, u) \right] + \lambda\alpha_T q_T [V(x, u) - V(x^+, u)]. \end{aligned}$$

Using the above three observations in (2.19), we have

$$\begin{aligned}
&Q(\overline{x}^+, u) - (1-\gamma)Q(\overline{x}, u) \\
\leq &\lambda\beta\left[(1 - \Lambda_T(1-\alpha_1) + \Lambda_T\alpha_1 p_1)V(x,u) - \alpha_T(1+p_T)V(x^+,u)\right] \\
&+ \lambda\alpha_T q_T[V(x,u) - V(x^+,u)] - \frac{\nu\beta}{2\gamma}\|\underline{x} - \overline{x}^+\|^2.
\end{aligned}$$

Comparing the above equation with (2.29), it now remains to show that

$$\alpha_T(1+p_T) = \Lambda_T\alpha_1 p_1 + 1 - \Lambda_T(1-\alpha_1).$$

By (2.14), the last relation in (2.28), and the fact that $\Lambda_1 = 1$, we have

$$\frac{\alpha_{t+1}p_{t+1}}{\Lambda_{t+1}} = \frac{\alpha_t p_t}{\Lambda_t} + \frac{\alpha_t}{\Lambda_t} = \ldots = \frac{\alpha_1 p_1}{\Lambda_1} + \sum_{i=1}^{t}\frac{\alpha_i}{\Lambda_i} = \alpha_1 p_1 + \frac{1 - \Lambda_t(1-\alpha_1)}{\Lambda_t}.$$

Using the second relation in (2.28) to the above equation, we have

$$\frac{\alpha_t(1+p_t)}{\Lambda_t} = \alpha_1 p_1 + \frac{1 - \Lambda_t(1-\alpha_1)}{\Lambda_t},$$

which implies $\alpha_t(1+p_t) = \Lambda_t\alpha_1 p_1 + 1 - \Lambda_t(1-\alpha_1)$ for any $1 \leq t \leq T$. □

With the help of the above proposition and Lemma 2.1, we are now ready to establish the convergence of the AGS method. Note that the following sequence will the used in the analysis of the AGS method:

$$\Gamma_k = \begin{cases} 1 & k = 1 \\ (1-\gamma_k)\Gamma_{k-1} & k > 1. \end{cases} \tag{2.30}$$

Theorem 2.1 *Suppose that* (2.18) *and* (2.28) *hold. If*

$$\gamma_1 = 1 \text{ and } \beta_k \geq \frac{L\gamma_k}{\nu}, \tag{2.31}$$

then

$$\begin{aligned}
&\phi(\overline{x}_k) - \phi(u) \\
\leq &\Gamma_k\sum_{i=1}^{k}\frac{\lambda_i\alpha_{T_i}(\beta_i(1+p_{T_i}) + q_{T_i})}{\Gamma_i}(V(x_{i-1},u) - V(x_i,u)),
\end{aligned} \tag{2.32}$$

where Γ_k is defined in (2.30).

Proof It follows from Proposition 2.1 that for all $u \in X$,

$$\begin{aligned}
&Q_k(\overline{x}_k, u) - (1-\gamma_k)Q_k(\overline{x}_{k-1}, u) \\
\leq &\lambda_k\alpha_{T_k}(\beta_k(1+p_{T_k}) + q_{T_k})(V(x_{k-1},u) - V(x_k,u)) - \frac{\nu\beta_k}{2\gamma_k}\|\overline{x}_k - \underline{x}_k\|^2.
\end{aligned}$$

Substituting the above bound to (2.10) in Lemma 2.1, and using (2.31), we have

$$\begin{aligned}
&\phi(\overline{x}_k) - \phi(u) \\
\leq &(1-\gamma_k)[\phi(\overline{x}_{k-1}) - \phi(u)] \\
&+ \lambda_k\alpha_{T_k}(\beta_k(1+p_{T_k}) + q_{T_k})(V(x_{k-1},u) - V(x_k,u)),
\end{aligned}$$

which, in view of Lemma 2.3 (with $c_k = \gamma_k$, $C_k = \Gamma_k$, and $\delta_k = \phi(\overline{x}_k) - \phi(u)$), then implies that

$$\begin{aligned}
&\phi(\overline{x}_k) - \phi(u) \\
\le &\Gamma_k \left[\frac{1-\gamma_1}{\Gamma_1}(\phi(\overline{x}_0) - \phi(u)) \right. \\
&\left. + \sum_{i=1}^{k} \frac{\lambda_i \alpha_{T_i}(\beta_i(1+p_{T_i}) + q_{T_i})}{\Gamma_i}(V(x_{i-1}, u) - V(x_i, u)) \right] \\
= &\Gamma_k \sum_{i=1}^{k} \frac{\lambda_i \alpha_{T_i}(\beta_i(1+p_{T_i}) + q_{T_i})}{\Gamma_i}(V(x_{i-1}, u) - V(x_i, u)),
\end{aligned}$$

where the last equality follows from the fact that $\gamma_1 = 1$ in (2.31). □

There are many possible selections of parameters that satisfy the assumptions of the above theorem. In the following corollaries we describe two different ways to specify the parameters of Algorithm 1 that lead to the optimal complexity bounds in terms of the number of gradient evaluations of ∇f and ∇h.

Corollary 2.1 *Consider problem (1.6) with the Lipschitz constants in (1.2) and (1.7) satisfing $M \ge L$. Suppose that the parameters of Algorithm 1 are set to*

$$\begin{aligned}
&\gamma_k = \frac{2}{k+1}, \ T_k \equiv T := \left\lceil \sqrt{\frac{M}{L}} \right\rceil, \\
&\lambda_k = \begin{cases} 1 & k = 1, \\ \dfrac{\gamma_k (T+1)(T+2)}{T(T+3)} & k > 1, \end{cases} \ \text{and } \beta_k = \frac{3L\gamma_k}{\nu k \lambda_k}.
\end{aligned} \tag{2.33}$$

Also assume that the parameters in the first call to the ProxAG procedure ($k = 1$) are set to

$$\alpha_t = \frac{2}{t+1}, \quad p_t = \frac{t-1}{2}, \ \text{and } q_t = \frac{6M}{\nu t}, \tag{2.34}$$

and the parameters in the remaining calls to the ProxAG procedure ($k > 1$) are set to

$$\alpha_t = \frac{2}{t+2}, \quad p_t = \frac{t}{2}, \ \text{and } q_t = \frac{6M}{\nu k(t+1)}. \tag{2.35}$$

Then the numbers of gradient evaluations of ∇f and ∇h performed by the AGS method to compute an ε-solution of (1.6) can be bounded by

$$N_f := \sqrt{\frac{30LV(x_0, x^*)}{\nu\varepsilon}} \tag{2.36}$$

and

$$N_h := \sqrt{\frac{30MV(x_0, x^*)}{\nu\varepsilon}} + \sqrt{\frac{30LV(x_0, x^*)}{\nu\varepsilon}} \tag{2.37}$$

respectively, where x^* is a solution to (1.6).

Proof Let us start with verification of (2.18), (2.28), and (2.31) for the purpose of applying Theorem 2.1. We will consider the first call to the ProxAG procedure ($k = 1$) and the remaining calls ($k > 1$) separately.

When $k = 1$, by (2.33) we have $\lambda_1 = \gamma_1 = 1$, and $\beta_1 = 3L/\nu$, hence (2.31) holds immediately. By (2.34) we can observe that $\Lambda_t = 2/(t(t+1))$ satisfies (2.13), and that

$$\frac{\alpha_t q_t}{\Lambda_t} \equiv \frac{6M}{\nu}, \ \text{and } \frac{\alpha_t(1+p_t)}{\Lambda_t} = \frac{t(t+1)}{2} = \frac{\alpha_{t+1}p_{t+1}}{\Lambda_{t+1}},$$

hence (2.28) holds. In addition, by (2.33) and (2.34) we have $\lambda = \gamma = 1$ and $\alpha_1 = 1$ in (2.18), and that

$$\beta p_t + q_t \geq q_t = \frac{6M}{\nu t} > \frac{2M}{\nu(t+1)} = \frac{\lambda M \alpha_t}{\nu}.$$

Therefore (2.18) also holds.

For the case when $k > 1$, we can observe from (2.35) that $\Lambda_t = 6/(t+1)(t+2)$ satisfies (2.13), $\alpha_t q_t/\Lambda_t \equiv 2M/(\nu k)$, and that

$$\frac{\alpha_t(1+p_t)}{\Lambda_t} = \frac{(t+1)(t+2)}{6} = \frac{\alpha_{t+1}p_{t+1}}{\Lambda_{t+1}}.$$

Therefore (2.28) holds. Also, from (2.33) and noting that $k, T \geq 1$, we have

$$\frac{3}{k} > \frac{3\gamma_k}{2} = \frac{3\lambda_k}{2}\left(1 - \frac{2}{(T+1)(T+2)}\right) \geq \frac{3\lambda_k}{2}\left(1 - \frac{2}{2\cdot 3}\right) = \lambda_k. \tag{2.38}$$

Applying the above relation to the definition of β_k in (2.33) we have (2.31). It now suffices to verify (2.18) in order to apply Theorem 2.1. Applying (2.33), (2.35), (2.38), and noting that $k \geq 2$ and that $\Lambda_T = 6/(T+1)(T+2)$ with $T \geq 1$, we can verify in (2.18) that

$$\begin{aligned}
&\lambda = \frac{\gamma(T+1)(T+2)}{T(T+3)} = \frac{2}{k+1}\left(1 + \frac{2}{T(T+3)}\right) \leq \frac{2}{3}\left(1 + \frac{2}{1\cdot 4}\right) = 1,\\
&\Lambda_T(1-\alpha_1) = \frac{2}{(T+1)(T+2)} = 1 - \frac{T(T+3)}{(T+1)(T+2)} = 1 - \frac{\gamma}{\lambda},\\
&\beta p_t + q_t > q_t = \frac{2M}{\nu(t+1)} \cdot \frac{3}{k} > \frac{2\lambda M}{\nu(t+1)} \geq \frac{\lambda M \alpha_t}{\nu}.
\end{aligned}$$

Therefore, the conditions in (2.18) are satisfied.

We are now ready to apply Theorem 2.1. In particular, noting that $\alpha_t(1+p_t) \equiv 1$ from (2.34) and (2.35), we obtain from (2.32) (with $u = x^*$) that

$$\phi(\overline{x}_k) - \phi^* \leq \Gamma_k \sum_{i=1}^{k} \xi_i (V(x_{i-1}, x^*) - V(x_i, x^*)), \tag{2.39}$$

where

$$\xi_i := \frac{\lambda_i(\beta_i + \alpha_{T_i} q_{T_i})}{\Gamma_i}, \tag{2.40}$$

Substituting (2.33) and (2.34) to (2.40), and noting that $\Gamma_i = 2/(i(i+1))$ by (2.30), we have

$$\begin{aligned}
\xi_1 =& \beta_1 + \alpha_T q_T = \frac{3L}{\nu} + \frac{12M}{\nu T(T+1)}, \text{ and}\\
\xi_i =& \frac{\lambda_i \beta_i}{\Gamma_i} + \frac{\lambda_i \alpha_{T_i} q_{T_i}}{\Gamma_i} = \frac{3L\gamma_i}{\nu i \Gamma_i} + \frac{\gamma_i}{\Gamma_i}\frac{(T_i+1)(T_i+2)}{T_i(T_i+3)}\frac{2}{T_i+2}\frac{6M}{\nu i(T_i+1)}\\
\equiv& \frac{3L}{\nu} + \frac{12M}{\nu T(T+3)}, \forall i > 1.
\end{aligned}$$

Applying the above two results regarding ξ_i to (2.39), and noting that $\xi_1 > \xi_2$, we have

$$\begin{aligned}
&\phi(\overline{x}_k) - \phi^* \\
\le&\Gamma_k \left[\xi_1(V(x_0,x^*) - V(x_1,x^*)) + \sum_{i=2}^{k} \xi_i(V(x_{i-1},x^*) - V(x_i,x^*))\right] \\
=&\Gamma_k \left[\xi_1(V(x_0,x^*) - V(x_1,x^*)) + \xi_2(V(x_1,x^*) - V(x_k,x^*))\right] \\
\le&\Gamma_k \xi_1 V(x_0,x^*) \\
=&\frac{2}{k(k+1)}\left(\frac{3L}{\nu} + \frac{12M}{\nu T(T+1)}\right) V(x_0,x^*) \\
\le&\frac{30L}{\nu k(k+1)} V(x_0,x^*),
\end{aligned}$$

where the last inequality is due to the fact that $T \ge \sqrt{M/L}$.

From the above inequality, the number of calls to the ProxAG procedure for computing an ε-solution of (1.6) is bounded by N_f in (2.36). This is also the bound for the number of gradient evaluations of ∇f. Moreover, the number of gradient evaluations of ∇h is bounded by

$$TN_f \le \left(\sqrt{\frac{M}{L}} + 1\right) N_f = \sqrt{\frac{30MV(x_0,x^*)}{\nu\varepsilon}} + \sqrt{\frac{30LV(x_0,x^*)}{\nu\varepsilon}} = N_h.$$

□

In the above corollary, the constant factors in (2.36) and (2.37) are both given by $\sqrt{30}$. In the following corollary, we provide a slightly different set of parameters for Algorithm 1 that results in a tighter constant factor for (2.36).

Corollary 2.2 *Consider problem (1.6) with the Lipschitz constants in (1.2) and (1.7) satisfing $M \ge L$. Suppose that the parameters in the first call to the ProxAG procedure ($k = 1$) are set to*

$$\alpha_t = \frac{2}{t+1},\ p_t = \frac{t-1}{2},\ \text{and } q_t = \frac{7LT(T+1)}{4\nu t}, \tag{2.41}$$

and that the parameters in the k-th call ($k > 1$) are set to

$$p_t \equiv p := \sqrt{\frac{M}{L}},\ \alpha_t \equiv \alpha := \frac{1}{p+1},\ \text{and } q_t \equiv 0. \tag{2.42}$$

If the other parameters in Algorithm 1 satisfy

$$\begin{aligned}
&\gamma_k = \frac{2}{k+1}, T_k := \begin{cases} \left\lceil \sqrt{\frac{8M}{7L}} \right\rceil, & k = 1 \\ \left\lceil \frac{\ln(3)}{-\ln(1-\alpha)} \right\rceil, & k > 1, \end{cases} \\
&\lambda_k := \begin{cases} 1, & k = 1 \\ \frac{\gamma_k}{1-(1-\alpha)^{T_k}}, & k > 1, \end{cases} \text{ and } \beta_k := \begin{cases} \frac{L}{\nu}, & k = 1 \\ \frac{9L\gamma_k}{2\nu k\lambda_k}, & k > 1, \end{cases}
\end{aligned} \tag{2.43}$$

where α is defined in (2.42), then the numbers of gradient evaluations of ∇f and ∇h performed by the AGS method to find an ε-solution to problem (1.6) can be bounded by

$$N_f := 3\sqrt{\frac{LV(x_0,x^*)}{\nu\varepsilon}} \tag{2.44}$$

and

$$\begin{aligned} N_h :=& (1+\ln 3) N_f \left(\sqrt{\frac{M}{L}} + 1 \right) \\ \leq& 7 \left(\sqrt{\frac{MV(x_0, x^*)}{\nu\varepsilon}} + \sqrt{\frac{LV(x_0, x^*)}{\nu\varepsilon}} \right), \end{aligned} \tag{2.45}$$

respectively.

Proof Let us verify (2.18), (2.31), and (2.28) first, so that we could apply Theorem 2.1. We consider the case when $k = 1$ first. By the definition of γ_k and β_k in (2.43), it is clear that (2.31) is satisfied when $k = 1$. Also, by (2.41) we have that $\Lambda_t = 2/(t(t+1))$ in (2.13),

$$\frac{\alpha_t q_t}{\Lambda_t} \equiv \frac{7LT_1(T_1+1)}{4\nu}, \text{ and } \frac{\alpha_t(1+p_t)}{\Lambda_t} = \frac{t(t+1)}{2} = \frac{\alpha_{t+1}p_{t+1}}{\Lambda_{t+1}},$$

hence (2.28) also holds. Moreover, by (2.41) and (2.43), we can verify in (2.18) that

$$\lambda = \gamma = 1, \Lambda_{T_1}(1-\alpha_1) = 0 = 1 - \frac{\gamma}{\lambda},$$

and

$$\beta p_t + q_t \geq q_t > \frac{7LT^2}{4\nu t} = \frac{8M}{4\nu t} > \frac{M\alpha_t}{\nu}.$$

Therefore the relations in (2.18) are all satisfied.

Now we consider the case when $k > 1$. By (2.13) and (2.42), we observe that $\Lambda_t = (1-\alpha)^{t-1}$ for all $t \geq 1$. Moreover, from the definition of T_k in (2.43), we can also observe that

$$(1-\alpha)^{T_k} \leq \frac{1}{3}.$$

Four relations can be derived based on the aforementioned two observations, (2.42), and (2.43). First,

$$\frac{\alpha_t q_t}{\Lambda_t} \equiv 0, \ \frac{\alpha_t(1+p_t)}{\Lambda_t} = \frac{1}{(1-\alpha)^{t-1}} = \frac{\alpha_{t+1}p_{t+1}}{\Lambda_{t+1}},$$

which verifies (2.28). Second,

$$\beta_k = \frac{9L(1-(1-\alpha)^{T_k})}{2\nu k} \geq \frac{3L}{\nu k} > \frac{L\gamma_k}{\nu},$$

which leads to (2.31). Third, noting that $k \geq 2$, we have

$$\frac{\gamma_k}{1-\Lambda_{T_k}(1-\alpha)} = \lambda_k = \frac{\gamma_k}{1-(1-\alpha)^{T_k}} \leq \frac{3\gamma_k}{2} = \frac{3}{k+1} \leq 1.$$

Fourth,

$$\begin{aligned} \frac{\nu\beta_k p}{\lambda_k M\alpha} =& \frac{9L\gamma_k p(p+1)}{2k\lambda_k^2 M} = \frac{9Lp(p+1)\left(1-(1-\alpha)^{T_k}\right)^2}{2k\gamma_k M} \\ =& \frac{9(k+1)}{4k} \cdot \left(\frac{Lp(p+1)}{M} \right) \cdot \left(1-(1-\alpha)^{T_k}\right)^2 \\ >& \frac{9}{4} \cdot 1 \cdot \frac{4}{9} = 1. \end{aligned}$$

The last two relations imply that (2.18) holds.

Summarizing the above discussions regarding both the cases $k = 1$ and $k > 1$, applying Theorem 2.1, and noting that $\alpha_t(1 + p_t) \equiv 1$, we have

$$\phi(\overline{x}_k) - \phi(u) \le \Gamma_k \sum_{i=1}^{k} \xi_i (V(x_{i-1}, u) - V(x_i, u)), \ \forall u \in X, \tag{2.46}$$

where

$$\xi_i := \frac{\lambda_i(\beta_i + \alpha_{T_i} q_{T_i})}{\Gamma_i}.$$

It should be observed from the definition of γ_k in (2.43) that $\Gamma_i := 2/(i(i+1))$ satisfies (2.30). Using this observation, applying (2.41), (2.42), and (2.43) to the above equation we have

$$\xi_1 = \beta_1 + \alpha_{T_1} q_{T_1} = \frac{L}{\nu} + \frac{7L}{2\nu} = \frac{9L}{2\nu}$$

and

$$\xi_i = \frac{\lambda_i \beta_i}{\Gamma_i} \equiv \frac{9L}{2\nu}, \ \forall i > 1.$$

Therefore, (2.46) becomes

$$\begin{aligned} \phi(\overline{x}_k) - \phi(u) \le & \frac{9L}{\nu k(k+1)} (V(x_0, u) - V(x_k, u)) \\ \le & \frac{9L}{\nu k(k+1)} V(x_0, u). \end{aligned} \tag{2.47}$$

Setting $u = x^*$ in the above inequality, we observe that the number of calls to the ProxAG procedure for computing an ε-solution of (1.6) is bounded by N_f in (2.44). This is also the bound for the number of gradient evaluations of ∇f. Moreover, by (2.42), (2.43), and (2.44) we conclude that the number of gradient evaluations of ∇h is bounded by

$$\begin{aligned} \sum_{k=1}^{N_f} T_k =& T_1 + \sum_{k=2}^{N_f} T_k \le \left(\sqrt{\frac{8M}{7L}} + 1 \right) + (N_f - 1) \left(\frac{\ln 3}{-\ln(1-\alpha)} + 1 \right) \\ \le & \left(\sqrt{\frac{8M}{7L}} + 1 \right) + (N_f - 1) \left(\frac{\ln 3}{\alpha} + 1 \right) \\ = & \left(\sqrt{\frac{8M}{7L}} + 1 \right) + (N_f - 1) \left(\left(\sqrt{\frac{M}{L}} + 1 \right) \ln 3 + 1 \right) \\ < & (1 + \ln 3) N_f \left(\sqrt{\frac{M}{L}} + 1 \right) \\ < & 7 \left(\sqrt{\frac{M V(x_0, x^*)}{\nu \varepsilon}} + \sqrt{\frac{L V(x_0, x^*)}{\nu \varepsilon}} \right). \end{aligned}$$

Here the second inequity is from the property of logarithm functions that $-\ln(1-\alpha) \ge \alpha$ for $\alpha \in [0, 1)$. □

Since $M \ge L$ in (1.2) and (1.7), the results obtained in Corollaries 2.1 and 2.2 indicate that the number of gradient evaluations of ∇f and ∇h that Algorithm 1 requires for computing an ε-solution of (1.6) can be bounded by $\mathcal{O}(\sqrt{L/\varepsilon})$ and $\mathcal{O}(\sqrt{M/\varepsilon})$, respectively. Such a result is particularly useful when M is significantly larger, e.g., $M = \mathcal{O}(L/\varepsilon)$, since the number of gradient evaluations of ∇f would not be affected at all by the large Lipschitz constant of the whole problem. It is interesting to compare the above result with the best known so-far complexity bound under the traditional black-box oracle assumption. If we treat problem (1.6) as a general smooth convex

optimization and study its oracle complexity, i.e., under the assumption that there exists an *oracle* that outputs $\nabla\phi(x)$ for any test point x (and $\nabla\phi(x)$ only), it has been shown that the number of calls to the oracle cannot be smaller than $\mathcal{O}(\sqrt{(L+M)/\varepsilon})$ for computing an ε-solution [24,27]. Under such "single oracle" assumption, the complexity bounds in terms of gradient evaluations of ∇f and ∇h are intertwined, and a larger Lipschitz constant M will result in more gradient evaluations of ∇f, even though there is no explicit relationship between ∇f and M. However, the results in Corollaries 2.1 and 2.2 suggest that we can study the oracle complexity of problem (1.6) based on the assumption of *two separate oracles*: one oracle $\mathcal{O}_f$ to compute ∇f for any test point x, and the other one $\mathcal{O}_h$ to compute $\nabla h(y)$ for any test point y. In particular, these two oracles do not have to be called at the same time, and hence it is possible to obtain separate complexity bounds $\mathcal{O}(\sqrt{L/\varepsilon})$ and $\mathcal{O}(\sqrt{M/\varepsilon})$ on the number of calls to $\mathcal{O}_f$ and $\mathcal{O}_h$, respectively.

We now consider a special case of (1.6) where f is strongly convex. More specifically, we assume that there exists $\mu > 0$ such that

$$\frac{\mu}{2}\|x-u\|^2 \le f(x) - l_f(u,x) \le \frac{L}{2}\|x-u\|^2, \ \forall x,u \in X. \tag{2.48}$$

Under the above assumption, we develop a multi-stage AGS algorithm that can skip computation of ∇f from time to time, and compute an ε-solution of (1.6) with

$$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}\right) \tag{2.49}$$

gradient evaluations of ∇f (see Alagorithm 2). It should be noted that, under the traditional black-box setting [24, 27] where one could only access $\nabla\psi(x)$ for each inquiry x, the number of evaluations of $\nabla\psi(x)$ required to compute an ε-solution is bounded by

$$\mathcal{O}\left(\sqrt{\frac{L+M}{\mu}}\log\frac{1}{\varepsilon}\right). \tag{2.50}$$

Algorithm 2 The multi-stage accelerated gradient sliding (M-AGS) algorithm

Choose $v_0 \in X$, accuracy ε, iteration limit N_0, and initial estimate Δ_0 such that $\phi(v_0) - \phi^* \le \Delta_0$.
for $s = 1, \ldots, S$ **do**
  Run the AGS algorithm with $x_0 = v_{s-1}$, $N = N_0$, and parameters in Corollary 2.2, and let $v_s = \overline{x}_N$.
end for
Output v_S.

Theorem 2.2 below describes the main convergence properties of the M-AGS algorithm.

Theorem 2.2 *Suppose that $M \ge L$ in (1.7) and (2.48), and that the prox-function $V(\cdot,\cdot)$ grows quadratically (i.e., (1.12) holds). If the parameters in Algorithm 2 are set to*

$$N_0 = 3\sqrt{\frac{2L}{\nu\mu}} \ \text{and} \ S = \log_2 \max\left\{\frac{\Delta_0}{\varepsilon}, 1\right\}, \tag{2.51}$$

then its output v_S must be an ε-solution of (1.1). Moreover, the total number of gradient evaluations of ∇f and ∇h performed by Algorithm 2 can be bounded by

$$N_f := 3\sqrt{\frac{2L}{\nu\mu}}\log_2 \max\left\{\frac{\Delta_0}{\varepsilon}, 1\right\} \tag{2.52}$$

and

$$\begin{aligned} N_h :=& (1+\ln 3)N_f \left(\sqrt{\frac{M}{L}}+1\right) \\ <& 9\left(\sqrt{\frac{L}{\nu\mu}}+\sqrt{\frac{M}{\nu\mu}}\right)\log_2 \max\left\{\frac{\Delta_0}{\varepsilon},1\right\}, \end{aligned} \tag{2.53}$$

respectively.

Proof With input $x_0 = v_{s-1}$ and $N = N_0$, we conclude from (2.47) in the proof of Corollary 2.2 (with $u = x^*$ a solution to (1.6)) that

$$\phi(\overline{x}_N) - \phi^* \le \frac{9L}{\nu N_0(N_0+1)} V(x_0, x^*) \le \frac{\mu}{2} V(x_0, x^*),$$

where the last inequality follows from (2.51). Using the facts that the input of the AGS algorithm is $x_0 = v_{s-1}$ and that the output is set to $v_s = \overline{x}_N$, and the relation (1.12), we conclude

$$\phi(v_s) - \phi^* \le \frac{\mu}{4}\|v_{s-1} - x^*\|^2 \le \frac{1}{2}(\phi(v_{s-1}) - \phi^*),$$

where the last inequality is due to the strong convexity of $\phi(\cdot)$. It then follows from the above relation, the definition of Δ_0 in Algorithm 2, and (2.51) that

$$\phi(v_S) - \phi^* \le \frac{1}{2^S}(\phi(v_0) - \phi^*) \le \frac{\Delta_0}{2^S} \le \varepsilon.$$

Comparing Algorithms 1 and 2, we can observe that the total number of gradient evaluations of ∇f in Algorithm 2 is bounded by $N_0 S$, and hence we have (2.52). Moreover, comparing (2.44) and (2.45) in Corollary 2.2, we conclude (2.53). □

In view of Theorem 2.2, the total number of gradient evaluations of ∇h required by the M-AGS algorithm to compute an ε-solution of (1.6) is the same as the traditional result (2.50). However, by skipping the gradient evaluations of ∇f from time to time in the M-AGS algorithm, the total number of gradient evaluations of ∇f is improved from (2.50) to (2.49). Such an improvement becomes more significant as the ratio M/L increases.

3 Application to composite bilinear saddle point problems

Our goal in this section is to show the advantages of the AGS method when applied to our motivating problem, i.e., the composite bilinear saddle point problem in (1.1). In particular, we show in Section 3.1 that the AGS algorithm can be used to solve (1.1) by incorporating the smoothing technique in [28] and derive new complexity bounds in terms of the number of gradient computations of ∇f and operator evaluations of K and K^T. Moreover, we demonstrate in Section 3.2 that even more significant saving on gradient computation of ∇f can be obtained when f is strongly convex in (1.1) by incorporating the multi-stage AGS method.

3.1 Saddle point problems

Our goal in this section is to extend the AGS algorithm from composite smooth optimization to nonsmooth optimization. By incorporating the smoothing technique in [28], we can apply AGS to solve the composite saddle point problem (1.1). Throughout this section, we assume that the dual feasible set Y in (1.1) is bounded, i.e., there exists $y_0 \in Y$ such that

$$\Omega := \max_{v\in Y} W(y_0, v)$$

is finite, where $W(\cdot,\cdot)$ is the prox-function associated with Y with modulus ω.

Let ψ_ρ be the smooth approximation of ψ defined in (1.3). It can be easily shown (see [28]) that

$$\psi_\rho(x) \le \psi(x) \le \psi_\rho(x) + \rho\Omega, \ \forall x \in X. \tag{3.1}$$

Therefore, if $\rho = \varepsilon/(2\Omega)$, then an $(\varepsilon/2)$-solution to (1.3) is also an ε-solution to (1.1). Moreover, it follows from Theorem 1 in [28] that problem (1.3) is given in the form of (1.6) (with $h(x) = h_\rho(x)$) and satisfies (1.7) with $M = \|K\|^2/(\rho\omega)$. Using these observations, we are ready to summarize the convergence properties of the AGS algorithm for solving problem (1.1).

Proposition 3.1 *Let $\varepsilon > 0$ be given and assume that $2\|K\|^2\Omega > \varepsilon\omega L$. If we apply the AGS method in Algorithm 1 to problem (1.3) (with $h = h_\rho$ and $\rho = \varepsilon/(2\Omega)$), in which the parameters are set to (2.41)–(2.43) with $M = \|K\|^2/(\rho\omega)$, then the total number of gradient evaluations of ∇f and linear operator evaluations of K (and K^T) in order to find an ε-solution of (1.1) can be bounded by*

$$N_f := 3\left(\sqrt{\frac{2LV(x_0,x^*)}{\nu\varepsilon}}\right) \tag{3.2}$$

and

$$N_K := 14\left(\sqrt{\frac{2LV(x_0,x^*)}{\nu\varepsilon}} + \frac{2\|K\|\sqrt{V(x_0,x^*)\Omega}}{\sqrt{\nu\omega}\varepsilon}\right), \tag{3.3}$$

respectively.

Proof By (3.1) we have $\psi_\rho^* \le \psi^*$ and $\psi(x) \le \psi_\rho(x) + \rho\Omega$ for all $x \in X$, and hence

$$\psi(x) - \psi^* \le \psi_\rho(x) - \psi_\rho^* + \rho\Omega, \ \forall x \in X.$$

Using the above relation and the fact that $\rho = \varepsilon/(2\Omega)$ we conclude that if $\psi_\rho(x) - \psi_\rho^* \le \varepsilon/2$, then x is an ε-solution to (1.1). To finish the proof, it suffices to consider the complexity of AGS for computing an $\varepsilon/2$-solution of (1.3). By Corollary 2.2, the total number of gradient evaluations of ∇f is bounded by (3.2). By Theorem 1 in [28], the evaluation of ∇h_ρ is equivalent to 2 evaluations of linear operators: one computation of form Kx for computing the maximizer $y^*(x)$ for problem (1.4), and one computation of form $K^T y^*(x)$ for computing $\nabla h_\rho(x)$. Using this observation, and substituting $M = \|K\|^2/(\rho\omega)$ to (2.45), we conclude (3.3). □

According to Proposition 3.1, the total number of gradient evaluations of ∇f and linear operator evaluations of both K and K^T are bounded by

$$\mathcal{O}\left(\sqrt{\frac{L}{\varepsilon}}\right) \tag{3.4}$$

and

$$\mathcal{O}\left(\sqrt{\frac{L}{\varepsilon}} + \frac{\|K\|}{\varepsilon}\right) \tag{3.5}$$

respectively, for computing an ε-solution of the saddle point problem (1.1). Therefore, if $L \le \mathcal{O}(\|K\|^2/\varepsilon)$, then the number of gradient evaluations of ∇f will not be affected by the dominating term $\mathcal{O}(\|K\|/\varepsilon)$. This result significantly improves the best known so-far complexity results for solving the bilinear saddle point problem (1.1) in [28] and [20]. Specifically, it improves the complexity regarding number of gradient computations of ∇f from $\mathcal{O}(1/\varepsilon)$ in [28] to $\mathcal{O}(1/\sqrt{\varepsilon})$, and also improves the complexity regarding operator evaluations involving K from $\mathcal{O}(1/\varepsilon^2)$ in [20] to $\mathcal{O}(1/\varepsilon)$.

3.2 Strongly convex composite saddle point problems

In this subsection, we still consider the SPP in (1.1), but assume that f is strongly convex (i.e., (2.48) holds). In this case, it has been shown previously in the literature that $\mathcal{O}(\|K\|/\sqrt{\varepsilon})$ first-order iterations, each one of them involving the computation of ∇f, and the evaluation of K and K^T, are needed in order to compute an ε-solution of (1.1) (e.g., [25]). However, we demonstrate in this subsection that the complexity with respect to the gradient evaluation of ∇f can be significantly improved from $\mathcal{O}(1/\sqrt{\varepsilon})$ to $\mathcal{O}(\log(1/\varepsilon))$.

Such an improvement can be achieved by properly restarting the AGS method applied to to solve a series of smooth optimization problem of form (1.3), in which the smoothing parameter ρ changes over time. The proposed multi-stage AGS algorithm with dynamic smoothing is stated in Algorithm 3.

Algorithm 3 The multi-stage AGS algorithm with dynamic smoothing

Choose $v_0 \in X$, accuracy ε, smoothing parameter ρ_0, iteration limit N_0, and initial estimate Δ_0 of (1.1) such that $\psi(v_0) - \psi^* \le \Delta_0$.
for $s = 1, \ldots, S$ **do**
Run the AGS algorithm to problem (1.3) with $\rho = 2^{-s/2}\rho_0$ (where $h = h_\rho$ in AGS). In the AGS algorithm, set $x_0 = v_{s-1}$, $N = N_0$, and parameters in Corollary 2.2, and let $v_s = \overline{x}_N$.
end for
Output v_S.

Theorem 3.1 describes the main convergence properties of Algorithm 3.

Theorem 3.1 *Let $\varepsilon > 0$ be given and suppose that the Lipschitz constant L in (2.48) satisfies*

$$\Omega\|K\|^2 \max\left\{\sqrt{\frac{15\Delta_0}{\varepsilon}}, 1\right\} \ge 2\omega\Delta_0 L.$$

Also assume that the prox-function $V(\cdot,\cdot)$ grows quadratically (i.e., (1.12) holds). If the parameters in Algorithm 3 are set to

$$N_0 = 3\sqrt{\frac{2L}{\nu\mu}},\ \ S = \log_2 \max\left\{\frac{15\Delta_0}{\varepsilon}, 1\right\},\ \ \text{and } \rho_0 = \frac{4\Delta_0}{\Omega 2^{S/2}}, \tag{3.6}$$

then the output v_S of this algorithm must be an ε-solution (1.1). Moreover, the total number of gradient evaluations of ∇f and operator evaluations involving K and K^T performed by Algorithm 3 can be bounded by

$$N_f := 3\sqrt{\frac{2L}{\nu\mu}} \log_2 \max\left\{\frac{15\Delta_0}{\varepsilon}, 1\right\} \tag{3.7}$$

and

$$N_K := 18\sqrt{\frac{L}{\nu\mu}} \log_2 \max\left\{\frac{15\Delta_0}{\varepsilon}, 1\right\} + \frac{56\sqrt{\Omega}\|K\|}{\sqrt{\mu\Delta_0\nu\omega}} \cdot \max\left\{\sqrt{\frac{15\Delta_0}{\varepsilon}}, 1\right\},$$

respectively.

Proof Suppose that x^* is an optimal solution to (1.1). By (2.47) in the proof of Corollary 2.2, in the s-th stage of Algorithm 3 (calling AGS with input $x_0 = v_{s-1}$, output $v_s = \overline{x}_N$, and iteration number $N = N_0$), we have

$$\begin{aligned}
&\psi_\rho(v_s) - \psi_\rho(x^*) = \psi_\rho(\overline{x}_N) - \psi_\rho(x^*) \\
\le &\frac{9L}{\nu N_0(N_0+1)} V(x_0, x^*) \le \frac{\mu}{2} V(x_0, x^*) \le \frac{\mu}{4}\|x_0 - x^*\|^2 = \frac{\mu}{4}\|v_{s-1} - x^*\|^2,
\end{aligned}$$

where the last two inequalities follow from (3.6) and (1.12), respectively. Moreover, by (3.1) we have $\psi(v_s) \leq \psi_\rho(v_s) + \rho\Omega$ and $\psi^* = \psi(x^*) \geq \psi_\rho(x^*)$, hence

$$\psi(v_s) - \psi^* \leq \psi_\rho(v_s) - \psi_\rho(x^*) + \rho\Omega.$$

Combing the above two equations and using the strong convexity of $\psi(\cdot)$, we have

$$\begin{aligned}\psi(v_s) - \psi^* \leq& \frac{\mu}{4}\|v_{s-1} - x^*\|^2 + \rho\Omega \\ \leq& \frac{1}{2}[\psi(v_{s-1}) - \psi^*] + \rho\Omega = \frac{1}{2}[\psi(v_{s-1}) - \psi^*] + 2^{-s/2}\rho_0\Omega,\end{aligned}$$

where the last equality is due to the selection of ρ in Algorithm 3. Reformulating the above relation as

$$2^s[\psi(v_s) - \psi^*] \leq 2^{s-1}[\psi(v_{s-1}) - \psi^*] + 2^{s/2}\rho_0\Omega,$$

and summing the above inequalities from $s = 1, \ldots, S$, we have

$$\begin{aligned}&2^S(\psi(v_S) - \psi^*) \\ \leq& \Delta_0 + \rho_0\Omega\sum_{s=1}^{S} 2^{s/2} = \Delta_0 + \rho_0\Omega\frac{\sqrt{2}(2^{S/2} - 1)}{\sqrt{2} - 1} < \Delta_0 + \frac{7}{2}\rho_0\Omega 2^{S/2} = 15\Delta_0,\end{aligned}$$

where the first inequality follows from the fact that $\psi(v_0) - \psi^* \leq \Delta_0$ and the last equality is due to (3.6). By (3.6) and the above result, we have $\psi(v_S) - \psi^* \leq \varepsilon$. Comparing the descriptions of Algorithms 1 and 3, we can clearly see that the total number of gradient evaluations of ∇f in Algorithm 3 is given $N_0 S$, hence we have (3.7).

To complete the proof it suffices to estimate the total number of operator evaluations involving K and K^T. By Theorem 1 in [28], in the s-th stage of Algorithm 3, the number of operator evaluations involving K is equivalent to twice the number of evaluations of ∇h_ρ in the AGS algorithm, which, in view of (2.45) in Corollary 2.2, is given by

$$\begin{aligned}&2(1 + \ln 3)N\left(\sqrt{\frac{M}{L}} + 1\right) \\ =&2(1 + \ln 3)N\left(\sqrt{\frac{\|K\|^2}{\rho\omega L}} + 1\right) = 2(1 + \ln 3)N_0\left(\sqrt{\frac{2^{s/2}\|K\|^2}{\rho_0\omega L}} + 1\right),\end{aligned}$$

where we used the relation $M = \|K\|^2/(\rho\omega)$ (see Section 3.1) in the first equality and relations $\rho = 2^{-s/2}\rho_0$ and $N = N_0$ from Algorithm 3 in the last equality. It then follows from the above result and (3.6) that the total number of operator evaluations involving K in Algorithm 3 can be bounded by

$$\begin{aligned}&\sum_{s=1}^{S} 2(1 + \ln 3)N_0\left(\sqrt{\frac{2^{s/2}\|K\|^2}{\rho_0\omega L}} + 1\right) \\ =&2(1 + \ln 3)N_0 S + \frac{2(1 + \ln 3)N_0\|K\|}{\sqrt{\rho_0\omega L}}\sum_{s=1}^{S} 2^{s/4} \\ =&2(1 + \ln 3)N_0 S + \frac{3\sqrt{2}(1 + \ln 3)\sqrt{\Omega}\|K\|2^{S/4}}{\sqrt{\mu\Delta_0\nu\omega}} \cdot \frac{2^{1/4}(2^{S/4} - 1)}{2^{1/4} - 1} \\ <&2(1 + \ln 3)N_0 S + \frac{56\sqrt{\Omega}\|K\|}{\sqrt{\mu\Delta_0\nu\omega}} \cdot 2^{S/2} \\ <&18\sqrt{\frac{L}{\nu\mu}}\log_2\max\left\{\frac{15\Delta_0}{\varepsilon}, 1\right\} + \frac{56\sqrt{\Omega}\|K\|}{\sqrt{\mu\Delta_0\nu\omega}} \cdot \max\left\{\sqrt{\frac{15\Delta_0}{\varepsilon}}, 1\right\}.\end{aligned}$$

□

By Theorem 3.1, the total number of operator evaluations involving K performed by Algorithm 3 to compute an ε-solution of (1.6) can be bounded by

$$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}+\frac{\|K\|}{\sqrt{\varepsilon}}\right),$$

which matches with the best-known complexity result (e.g., [25]). However, the total number of gradient evaluations of ∇f is now bounded by

$$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}\right),$$

which drastically improves existing results from $\mathcal{O}(1/\sqrt{\varepsilon})$ to $\mathcal{O}(\log(1/\varepsilon))$.

4 Numerical experiments

In this section, we present some preliminary experimental results on the proposed AGS algorithm. The algorithms for all the experiments are implemented in MATLAB R2016a, running on a computer with 3.6 GHz Intel i7-4790 CPU and 32GB RAM. The parameters of Algorithm 1 are set to Corollary 2.2 and Proposition 3.1 for solving composite smooth problems and bilinear saddle point problems, respectively.

4.1 Smooth optimization

Our first experiment is conducted on a portfolio selection problem, which can be formulated as a quadratic programming problem

$$\min_{x\in\Delta^n}\phi(x):=x^T(A^T\mathcal{F}A+\mathcal{D})x\quad\text{s.t. }\ b^Tx\geq\eta, \tag{4.1}$$

where $\Delta^n:=\{x\in\mathbb{R}^n|\sum_{i=1}^n x_i=1,x_i\geq 0,i=1,\ldots,n\}$. The above quadratic programming problem describes minimum variance portfolio selection strategy in a market with n trading assets and m factors that drive the market. In particular, we are assuming a market return model (see [11])

$$q=b+A^Tf+\varepsilon,$$

where $q\in\mathbb{R}^n$ describes the random return with mean $b\in\mathbb{R}^n$, $f\in\mathbb{R}^m$ is a normally distributed vector with distribution $f\sim N(0,\mathcal{F})$ that describes the factors driving the market, $A\in\mathbb{R}^{m\times n}$ is the matrix of factor loadings of the n assets, and $\varepsilon\sim N(0,\mathcal{D})$ is the random vector of residual returns. The return of portfolio x now follows the distribution

$$q^Tx\sim N(b^Tx,x^T(A^T\mathcal{F}A+\mathcal{D})x),$$

and problem (4.1) describes the objective of minimizing the risk (in terms of variance) while obtaining expected return of at least η. It can be easily seen that Problem (4.1) is a special case of (1.6) with

$$f(x)=x^T\mathcal{D}x,h(x)=x^T(A^T\mathcal{F}A)x,\ X=\left\{x\in\Delta^n\ \middle|\ b^Tx\geq\eta\right\},$$
$$M=\lambda_{max}(A^T\mathcal{F}A),\ \text{and}\ L=\lambda_{max}(\mathcal{D}).$$

Here $\lambda_{max}(\cdot)$ denotes the maximum eigenvalue. It should be noted that in practice we have $m<n$ and the eigenvalues of $\mathcal{D}$ are much smaller than that of $A^T\mathcal{F}A$. Consequently, the computational cost for gradient evaluation of ∇f is more expensive than that of ∇h, and the Lipschitz constants L in (1.2) and M in (1.7) satisfy $L<M$.

To generate the datasets for this experiment, first we fix $n = 5000$, $\eta = 1$, choose m from $\{2^4, 2^5, \ldots, 2^9\}$, and generate b, A, and $\mathcal{F}$ randomly. Here each component of b is generated uniformly in $[0, 5]$, each component of A is generated uniformly in $[0, 1]$, and $\mathcal{F} = B^T B$ where B is a $\lceil m/2 \rceil \times m$ matrix whose components are generated from standard normal distribution. Then, we estimate M from the relation $M = \lambda_{max}(A^T \mathcal{F} A)$, choose L from $\{2^{-2}M, 2^{-3}M, \ldots, 2^{-15}M\}$, and set $\mathcal{D} = L(C^T C)/\lambda_{max}(C^T C)$, where C is a 2500 by 5000 matrix whose components are generated from the standard normal distribution. With such setting of $\mathcal{D}$, we have $\lambda_{max}(D) = L$. In order to demonstrate the efficiency of the AGS algorithm, we compare it with Nesterov's accelerated gradient method (NEST) in [27]. For both algorithms, we set the prox-function to the entropy function $V(x, u) = \sum_{i=1}^{n} u^{(i)} \ln(u^{(i)}/x^{(i)})$, where $u^{(i)}$ denotes the i-th component of u. Observe that the computational costs per iteration are different for NEST and AGS, since NEST evaluates both ∇f and ∇h in each iteration, while AGS can skip the evaluation of ∇f from time to time. In order to have a fair comparison between these two algorithms, we first run NEST for 300 iterations, and then AGS for the same amount of CPU time as NEST. In Figure 4.1, we plot the ratio of the objective values obtained by NEST of AGS in this manner. If the ratio is less than 1 (indicated by red cross in the figure), then the performance of NEST is better than that of the AGS, and if the ratio is greater than 1 (indicated by blue round), then AGS outperforms NEST. It should be noted that we plot the ratio rather than the difference of the objective values obtained by these algorithms, mainly because these objective values for difference instances are quite different. We can observe from Figure 4.1 that for most of the choices of m and L, AGS outperforms NEST in terms of objective value. In particular, as M/L increases and m decreases, the difference on the performance of AGS and NEST becomes more and more significant. Therefore, we can conclude that AGS performs better than NEST when either the difference between M and L is larger or the computational cost for evaluating ∇h is cheaper. Such observations are consistent with our theoretical complexity analysis regarding AGS and NEST.

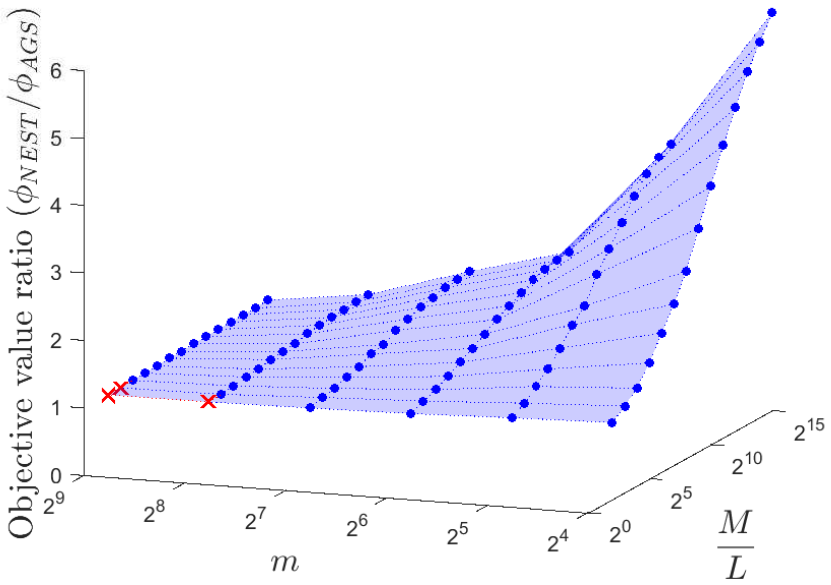

Fig. 4.1 Ratio of objective values of AGS and NEST in terms of different choices of dimension m and ratio M/L, after running the same amount of CPU time. Here ϕ_{AGS} and ϕ_{NEST} are the objective values corresponding to approximated solutions obtained by AGS and NEST within the same amount of CPU time. Cross markers in red imply that $\phi_{NEST} < \phi_{AGS}$, i.e., NEST outperforms AGS, while round markers in blue indicate that AGS outperforms NEST.

In addition to Figure 4.1, we also report in Tables 4.1 and 4.2 the numbers of gradient evaluations of ∇f and ∇h performed by the AGS method with respect to different choices of dimension m and ratio M/L. As

mentioned earlier, the numbers of gradient evaluations of ∇f and ∇h in 300 iterations of NEST are both 300. Several remarks are in place regarding the results obtained in these two tables. First, during the same amount of CPU time, the AGS is able to perform more gradient evaluations of ∇h by skipping the computation of ∇f. Noting that the Lipschitz constants of ∇h and ∇f satisfy $M > L$, by the complexity bounds (1.8), the increased number of gradient evaluations of ∇h results in lower objective function value of AGS. The advantage of AGS over NEST in terms of objective function value becomes more significant as the ratio M/L increases. Second, the lower computational cost we have for gradient evaluation of ∇h, the more gradient evaluations of ∇h we can perform at each time when skipping gradient evaluation of ∇f. Therefore, we can observe in Table 4.1 that the reduction of dimension m leads to more evaluations of ∇h, and more significant performance improvment of AGS over NEST. Finally, it should be noted that AGS requires more evaluations of ∇h in order to obtain the same level of accuracy as NEST in terms of objective function value. In particular, we can observe from the case with $m = 512$ in Table 4.1 and the case with $M/L = 2^2$ in Table 4.2 that AGS requires approximately triple amount of gradient evaluations of ∇h in order to obtain the same objective value as NEST. One plausible explanation is that the estimate of M/L in Corollary 2.2 is conservative, resulting in a larger number of inner iterations T_k in (2.43). However, when m is small or when the estimated ratio M/L is high, AGS can perform much more numbers of gradient evaluations of ∇h to overcome the aforementioned disadvantage. It would be interesting to develop a scheme that provides more accurate estimate of the ratio M/L, possibly through some line search procedures.

Table 4.1 Numbers of gradient evaluations of ∇f and ∇h performed by the AGS method for solving (4.1) with $M/L = 1024$, after running the same amount of CPU time as 300 iterations of NEST. Here ϕ_{AGS} and ϕ_{NEST} are the objective values corresponding to the approximated solutions obtained by AGS and NEST, respectively.

m	# AGS evaluations of ∇f	# AGS evaluations of ∇h	ϕ_{NEST}/ϕ_{AGS}
16	104	3743	382.5%
32	100	3599	278.6%
64	95	3419	183.3%
128	65	2339	152.8%
256	42	1499	120.1%
512	27	936	104.8%

Table 4.2 Numbers of gradient evaluations of ∇f and ∇h performed by the AGS method for solving (4.1) with $m = 64$, after running the same amount of CPU time as 300 iterations of NEST. Here ϕ_{AGS} and ϕ_{NEST} are the objective values corresponding to the approximated solutions obtained by AGS and NEST, respectively.

M/L	# AGS evaluations of ∇f	# AGS evaluations of ∇h	ϕ_{NEST}/ϕ_{AGS}
2^{15}	23	4471	212.5%
2^{14}	31	4327	210.5%
2^{13}	41	4097	206.5%
2^{12}	57	4038	201.6%
2^{11}	72	3648	192.4%
2^{10}	95	3419	183.3%
2^{9}	114	2961	173.3%
2^{8}	143	2698	161.7%
2^{7}	164	2132	150.5%
2^{6}	186	1859	140.1%
2^{5}	210	1470	129.2%
2^{4}	225	1125	120.0%
2^{3}	258	1032	112.9%
2^{2}	253	759	104.5%

4.2 Image reconstruction

In this subsection, we consider the following total-variation (TV) regularized image reconstruction problem:

$$\min_{x\in\mathbb{R}^n} \psi(x) := \frac{1}{2}\|Ax-b\|^2 + \eta\|Dx\|_{2,1}. \tag{4.2}$$

Here $x \in \mathbb{R}^n$ is the n-vector form of a two-dimensional image to be reconstructed, $\|Dx\|_{2,1}$ is the discrete form of the TV semi-norm where D is the finite difference operator, A is a measurement matrix describing the physics of data acquisition, and b is the observed data. It should be noted that problem (4.2) is equivalent to

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|^2 + \max_{y\in Y} \eta\langle Dx, y\rangle,$$

where $Y := \{y \in \mathbb{R}^{2n} : \|y\|_{2,\infty} := \max_{i=1,\ldots,n} \|(y^{(2i-1)}, y^{(2i)})^T\|_2 \le 1\}$. The above form can be viewed as a special case of the bilinear SPP (1.1) with

$$f(x) := \frac{1}{2}\|Ax-b\|^2, K := \eta D, \text{ and } J(y) \equiv 0,$$

and the associated constants are $L = \lambda_{max}(A^TA)$ and $\|K\| = \eta\sqrt{8}$ (see, e.g., [4]). Therefore, as discussed in Section 3.1, such problem can be solved by AGS after incorporating the smoothing technique in [28].

In this experiment, the dimension of $A \in \mathbb{R}^{m\times n}$ is set to $m = \lceil n/3 \rceil$. Each component of A is generated from a Bernoulli distribution, namely, it takes equal probability for the values $1/\sqrt{m}$ and $-1/\sqrt{m}$ respectively. We generate b from a ground truth image x_{true} with $b = Ax_{true} + \varepsilon$, where $\varepsilon \sim N(0, 0.001I_n)$. Two ground truth images x_{true} are used in the experiment, namely, the 256 by 256 ($n = 65536$) image "Cameraman" and the 135 by 198 ($n = 26730$) image "Onion". Both of them are built-in test images in the MATLAB image processing toolbox. We compare the performance of AGS and NEST for each test image with different smoothing parameter ρ in (1.4), and TV regularization parameter η in (4.2). For both algorithms, the prox-functions $V(x,u)$ and $W(y,v)$ are set to Euclidean distances $\|x-u\|_2^2/2$ and $\|y-v\|_2^2/2$ respectively. In order to perform a fair comparison, we run NEST for 200 iterations first, and then run AGS with the same amount of CPU time.

Tables 4.3 and 4.4 show the comparison between AGS and NEST in terms of gradient evaluations of ∇f, operator evaluations of K and K^T, and objective values (4.2). It should be noted that in 200 iterations of the NEST algorithm, the number of gradient evaluations of ∇f and operator evaluations of K and K^T are given by 200 and 400, respectively. We can make a few observations about the results reported in these tables. First, by skipping gradient evaluations of ∇f, AGS is able to perform more operator evaluation of K and K^T during the same amount of CPU time. Noting the complexity bounds (3.4) and (3.5), we can observe that the extra amount of operator evaluations K and K^T can possibly result in better approximate solutions obtained by CGS in terms of objective values. It should be noted that in problem (4.2), A is a dense matrix while D is a sparse matrix. Therefore, a very large number of extra evaluations of K and K^T can be performed for each skipped gradient evaluation of ∇f. Second, for the smooth approximation problem (1.3), the Lipschitz constant M of h_ρ is given by $M = \|K\|^2/\rho\omega$. Therefore, for the cases with ρ being fixed, larger values of ρ result in larger norm $\|K\|$, and consequently larger Lipschitz constant M. Moreover, for the cases when η is fixed, smaller values of ρ also lead to larger Lipschitz constant M. For both cases, as the ratio of M/L increases, we would skip more and more gradient evaluations of ∇f, and allocate more CPU time for operator evaluations of K and K^T, which results in more significant performance improvement of AGS over NEST. Such observations are also consistent with our previous theoretical complexity analysis regarding AGS and NEST for solving composite bilinear saddle point problems.

Table 4.3 Numbers of gradient evaluations of ∇f and ∇h performed by the AGS method for solving (4.2) with ground truth image "Cameraman", after running the same amount of CPU time as 200 iterations of NEST. Here ψ_{AGS} and ψ_{NEST} are the objective values of (4.2) corresponding to approximated solutions obtained by AGS and NEST, respectively.

Problem	# AGS evaluations of ∇f	# AGS evaluations of K and K^T	ψ_{AGS}	ψ_{NEST}
$\eta = 1, \quad \rho = 10^{-5}$	52	37416	723.8	8803.1
$\eta = 10^{-1}, \quad \rho = 10^{-5}$	173	12728	183.2	2033.5
$\eta = 10^{-2}, \quad \rho = 10^{-5}$	198	1970	27.2	38.3
$\eta = 10^{-1}, \quad \rho = 10^{-7}$	51	36514	190.2	8582.1
$\eta = 10^{-1}, \quad \rho = 10^{-6}$	118	27100	183.2	6255.6
$\eta = 10^{-1}, \quad \rho = 10^{-5}$	173	12728	183.2	2033.5
$\eta = 10^{-1}, \quad \rho = 10^{-4}$	192	4586	183.8	267.2
$\eta = 10^{-1}, \quad \rho = 10^{-3}$	201	2000	190.4	191.2
$\eta = 10^{-1}, \quad \rho = 10^{-2}$	199	794	254.2	254.2

Table 4.4 Numbers of gradient evaluations of ∇f and ∇h performed by the AGS method for solving (4.2), after running the same amount of CPU time as 200 iterations of NEST. Here ψ_{AGS} and ψ_{NEST} are the objective values of (4.2) corresponding to approximated solutions obtained by AGS and NEST, respectively.

Problem	# AGS evaluations of ∇f	# AGS evaluations of K and K^T	ψ_{AGS}	ψ_{NEST}
$\eta = 1, \quad \rho = 10^{-5}$	37	26312	295.6	2380.3
$\eta = 10^{-1}, \quad \rho = 10^{-5}$	149	10952	52.6	608.5
$\eta = 10^{-2}, \quad \rho = 10^{-5}$	193	1920	6.9	10.6
$\eta = 10^{-1}, \quad \rho = 10^{-7}$	62	44344	52.9	2325.5
$\eta = 10^{-1}, \quad \rho = 10^{-6}$	102	23380	52.6	1735.1
$\eta = 10^{-1}, \quad \rho = 10^{-5}$	149	10952	52.6	608.5
$\eta = 10^{-1}, \quad \rho = 10^{-4}$	174	4154	52.8	70.0
$\eta = 10^{-1}, \quad \rho = 10^{-3}$	192	1910	54.5	54.7
$\eta = 10^{-1}, \quad \rho = 10^{-2}$	198	790	68.6	68.6

5 Conclusion

We propose an accelerated gradient sliding (AGS) method for solving certain classes of structured convex optimization. The main feature of the proposed AGS method is that it could skip gradient computations of a smooth component in the objective function from time to time, while still maintaining the overall optimal rate of convergence for these probems. In particular, for minimizing the summation of two smooth convex functions, the AGS method can skip the gradient computation of the function with a smaller Lipschitz constant, resulting in sharper complexity results than the best known so-far complexity bound under the traditional black-box assumption. Moreover, for solving a class of bilinear saddle-point problem, by applying the AGS algorithm to solve its smooth approximation, we show that the number of gradient evaluations of the smooth component may be reduced to $\mathcal{O}(1/\sqrt{\varepsilon})$, which improves the previous $\mathcal{O}(1/\varepsilon)$ complexity bound in the literature. More significant savings on gradient computations can be obtained when the objective function is strongly convex, with the number of gradient evaluations being reduced further to $\mathcal{O}(\log(1/\varepsilon))$. Numerical experiments further confirm the potential advantages of these new optimization schemes for solving structured convex optimization problems.

References

1. Auslender, A., Teboulle, M.: Interior gradient and proximal methods for convex and conic optimization. SIAM Journal on Optimization **16**(3), 697–725 (2006)
2. Becker, S., Bobin, J., Candès, E.: NESTA: a fast and accurate first-order method for sparse recovery. SIAM Journal on Imaging Sciences **4**(1), 1–39 (2011)
3. Bregman, L.M.: The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. USSR computational mathematics and mathematical physics **7**(3), 200–217 (1967)

4. Chambolle, A.: An algorithm for total variation minimization and applications. Journal of Mathematical Imaging and Vision **20**(1), 89–97 (2004)
5. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of Mathematical Imaging and Vision **40**(1), 120–145 (2011)
6. Chen, Y., Lan, G., Ouyang, Y.: Accelerated schemes for a class of variational inequalities. arXiv preprint arXiv:1403.4164 (2014)
7. Chen, Y., Lan, G., Ouyang, Y.: Optimal primal-dual methods for a class of saddle point problems. SIAM Journal on Optimization **24**(4), 1779–1814 (2014)
8. d'Aspremont, A.: Smooth optimization with approximate gradient. SIAM Journal on Optimization **19**(3), 1171–1183 (2008)
9. Esser, E., Zhang, X., Chan, T.: A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. SIAM Journal on Imaging Sciences **3**(4), 1015–1046 (2010)
10. Ghadimi, S., Lan, G.: Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. SIAM Journal on Optimization **22**(4), 1469–1492 (2012)
11. Goldfarb, D., Iyengar, G.: Robust portfolio selection problems. Mathematics of Operations Research **28**(1), 1–38 (2003)
12. He, B., Yuan, X.: Convergence analysis of primal-dual algorithms for a saddle-point problem: From contraction perspective. SIAM Journal on Imaging Sciences **5**(1), 119–149 (2012)
13. He, B., Yuan, X.: On the O(1/n) convergence rate of the Douglas-Rdachford alternating direction method. SIAM Journal on Numerical Analysis **50**(2), 700–709 (2012)
14. He, N., Juditsky, A., Nemirovski, A.: Mirror prox algorithm for multi-term composite minimization and alternating directions. arXiv preprint arXiv:1311.1098 (2013)
15. He, Y., Monteiro, R.D.: Accelerating block-decomposition first-order methods for solving generalized saddle-point and nash equilibrium problems. Optimization-online preprint (2013)
16. He, Y., Monteiro, R.D.: An accelerated hpe-type algorithm for a class of composite convex-concave saddle-point problems. Submitted to SIAM Journal on Optimization (2014)
17. Hoda, S., Gilpin, A., Pena, J., Sandholm, T.: Smoothing techniques for computing nash equilibria of sequential games. Mathematics of Operations Research **35**(2), 494–512 (2010)
18. Juditsky, A., Nemirovski, A., Tauvel, C.: Solving variational inequalities with stochastic mirror-prox algorithm. Stochastic Systems **1**, 17–58 (2011)
19. Lan, G.: Bundle-level type methods uniformly optimal for smooth and nonsmooth convex optimization. Mathematical Programming pp. 1–45 (2013)
20. Lan, G.: Gradient sliding for composite optimization. Mathematical Programming pp. 1–35 (2015)
21. Lan, G., Lu, Z., Monteiro, R.D.: Primal-dual first-order methods with $\mathcal{O}(1/\varepsilon)$ iteration-complexity for cone programming. Mathematical Programming **126**(1), 1–29 (2011)
22. Monteiro, R.D., Svaiter, B.F.: Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers. SIAM Journal on Optimization **23**(1), 475–507 (2013)
23. Nemirovski, A.: Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. SIAM Journal on Optimization **15**(1), 229–251 (2004)
24. Nemirovski, A., Yudin, D.: Problem complexity and method efficiency in optimization. Wiley-Interscience Series in Discrete Mathematics. John Wiley, XV (1983)
25. Nesterov, Y.: Excessive gap technique in nonsmooth convex minimization. SIAM Journal on Optimization **16**(1), 235–249 (2005)
26. Nesterov, Y.E.: A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. Doklady AN SSSR **269**, 543–547 (1983)
27. Nesterov, Y.E.: Introductory Lectures on Convex Optimization: A Basic Course. Kluwer Academic Publishers, Massachusetts (2004)
28. Nesterov, Y.E.: Smooth minimization of nonsmooth functions. Mathematical Programming **103**, 127–152 (2005)
29. Ouyang, H., He, N., Tran, L., Gray, A.G.: Stochastic alternating direction method of multipliers. In: Proceedings of the 30th International Conference on Machine Learning (ICML-13), pp. 80–88 (2013)
30. Ouyang, Y., Chen, Y., Lan, G., Eduardo Pasiliao, J.: An accelerated linearized alternating direction method of multipliers. SIAM Journal on Imaging Sciences **8**(1), 644–681 (2015)
31. Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization. submitted to SIAM Journal on Optimization (2008)
32. Zhu, M., Chan, T.: An efficient primal-dual hybrid gradient algorithm for total variation image restoration. UCLA CAM Report pp. 08–34 (2008)