Journal of Scientific Computing (2019) 79:1854–1881
https://doi.org/10.1007/s10915-019-00915-4

Generalized Uniformly Optimal Methods for Nonlinear Programming

Saeed Ghadimi[1] · Guanghui Lan[2] · Hongchao Zhang[3]

Received: 14 September 2017 / Revised: 20 June 2018 / Accepted: 21 January 2019 /
Published online: 14 February 2019

Abstract
Uniformly optimal convex programming algorithms have been designed to achieve the optimal complexity bounds for convex optimization problems regardless of the level of smoothness of the objective function. In this paper, we present a generic framework to extend such existing algorithms to solve more general nonlinear, possibly nonconvex, optimization problems. The basic idea is to incorporate a local search step (gradient descent or Quasi-Newton iteration) into the uniformly optimal convex programming methods, and then enforce a monotone decreasing property of the function values computed along the trajectory. While optimal methods for nonconvex programming are not generally known, algorithms of these types will achieve the best known complexity for nonconvex problems, and the optimal complexity for convex ones without requiring any problem parameters. As a consequence, we can have a unified treatment for a general class of nonlinear programming problems regardless of their convexity and smoothness level. In particular, we show that the accelerated gradient and level methods, both originally designed for solving convex optimization problems only, can be used for solving both convex and nonconvex problems uniformly. In a similar vein, we show that some well-studied techniques for nonlinear programming, e.g., Quasi-Newton iteration, can be embedded into optimal convex optimization algorithms to possibly further enhance their numerical performance. Our theoretical and algorithmic developments are complemented by some promising numerical results obtained for solving a few important nonconvex and nonlinear data analysis problems in the literature.

Keywords Nonconvex optimization · Uniformly optimal methods · Parameter free methods · Quasi-Newton methods · Accelerated gradient methods · Accelerated level methods

Mathematics Subject Classification 90C25 · 90C06 · 90C26 · 49M37

This research was partially supported by NSF Grants CMMI-1254446, CMMI-1537414, DMS-1319050, DMS-1522654, DMS-1819161 and ONR Grant N00014-13-1-0036. This paper was first released on ArXiv in August, 2015 (arXiv:1508.07384).

Extended author information available on the last page of the article

1 Introduction

In this paper, we consider the following nonlinear programming problem

$$\Psi^* := \min_{x \in X}\{\Psi(x) := f(x) + \mathcal{X}(x)\}, \tag{1.1}$$

where $X \subseteq \mathbb{R}^n$ is a closed convex set, $f : X \to \mathbb{R}$ is possibly nonconvex, and $\mathcal{X}$ is a simple but possibly non-differentiable convex function with known structure (e.g. $\mathcal{X}(x) = \|x\|_1$ or $\mathcal{X}$ even vanishes). Moreover, we assume that f has Hölder continuous gradient on X i.e., there exists $H > 0$ and $\nu \in [0, 1]$ such that

$$\|f'(y) - f'(x)\| \le H\|y - x\|^{\nu} \quad \forall x, y \in X, \tag{1.2}$$

where $f'(x)$ is the gradient of f at x and $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^n$. The above assumption in (1.2) covers a wide range class of objective functions, including smooth functions (with $\nu = 1$), weakly smooth functions (with $\nu \in (0, 1)$), and nonsmooth convex functions with bounded subgradients (with $\nu = 0$).

The complexity of solving (1.1) has been well-understood under the convex setting, i.e., when f is convex. According to the classic complexity theory by Nemirovski and Yudin [21], if f is a general convex function with bounded subgradients (i.e., $\nu = 0$), then the number of subgradient evaluations of f required to find a solution $\bar{x} \in X$ such that $\Psi(\bar{x}) - \Psi^* \le \epsilon$ cannot be smaller than $\mathcal{O}(1/\epsilon^2)$ when n is sufficiently large. Here Ψ^* denotes the optimal value of (1.1). Such a lower complexity bound can be achieved by different first-order methods, including the subgradient and mirror descent methods in [21], and the bundle-level type methods in [4,18]. Moreover, if f is a smooth convex function with Lipschitz continuous gradients (i.e., $\nu = 1$), then the number of gradient evaluations of f cannot be smaller than $\mathcal{O}(1/\sqrt{\epsilon})$ for sufficiently large n. Such a lower complexity bound can be achieved by the well-known Nesterov's accelerated gradient (AG) method, which was originally developed in [22] for the smooth case with $\mathcal{X} = 0$, and recently extended for an emerging class of composite problems with a relatively simple nonsmooth term $\mathcal{X}$ [2,24,29].

While traditionally different classes of convex optimization problems were solved by using different algorithms, the last few years have seen a growing interest in the development of unified algorithms that can achieve the optimal complexity for solving different classes of convex problems, preferably without requiring the input of any problem parameters. Lan showed in [14] that Nesterov's AG method in [22,23] can achieve the optimal complexity for solving not only smooth, but also general nonsmooth optimization problems (i.e., $\nu = 0$ in (1.2)) by introducing a novel stepsize policy and some new convergence analysis techniques. Devolder, Glineur and Nesterov [28] further generalized this development and showed that the AG method can exhibit the optimal $\mathcal{O}(1/\epsilon^{\frac{2}{1+3\nu}})$ complexity for solving weakly smooth problems. These methods in [14,28] still require the input of problem parameters like ν and H, and even the iteration limit N. In a different research line, Lan [15] generalized the bundle-level type methods, originally designed for nonsmooth problems, for both smooth and weakly smooth problems. He showed that these accelerated level methods are uniformly optimal for convex optimization in the sense that they can achieve the optimal complexity bounds without requiring the input of any problem parameters and the iteration limit. Simplified variants of these methods were also proposed in [7] for solving ball-constrained and unconstrained convex optimization problems. Recently, Nesterov [25] presented another uniformly optimal method, namely, the universal accelerated gradient method for nonsmooth and (weakly) smooth convex programming. This method only needs the target accuracy as an input, which, similar to those in [7,15], completely removes the need of inputting problem parameters in [14,28].

In general, all the above-mentioned methods require the convexity on the objective function to establish their convergence results. When f is possibly nonconvex, a different termination criterion according to the projected gradient $g_{X,k}$ (see e.g., (2.16)) is often employed to analyze the complexity of the solution methods. While there is no known lower iteration complexity bound for first-order methods to solve the problem (1.1), the (projected) gradient-type methods [6,12,23] achieve the best-known iteration complexity $\mathcal{O}(1/\epsilon)$ to find a solution such that $\|g_{X,k}\|^2 \le \epsilon$ when f in (1.1) has Lipschitz continuous gradient. Recently, Ghadimi and Lan [11] generalized Nesterov's AG method to solve this class of nonconvex nonlinear optimization problems. They showed that this generalized AG method can not only achieve the best-known $\mathcal{O}(1/\epsilon)$ iteration complexity for finding approximate stationary points for nonconvex problems, but also exhibit the optimal iteration complexity if the objective function turns out to be convex. However, in oder to apply this method, we need to assume that all the generated iterates lie in a bounded set and that the gradients of f are be Lipschitz continuous, and also requires the input of a few problem parameters a priori. Our main goal of this paper is to understand whether we can generalize some of the aforementioned uniformly optimal methods to solve a broader class of nonlinear programming given in (1.1), where function f could be nonconvex and only weakly smooth or nonsmooth (see (1.2)). In addition to these theoretical aspects, our study has also been motivated by the following applications.

– In many machine learning problems, the regularized loss function in the objective is given as a summation of convex and nonconvex terms (see e.g., [8,19,20]). A unified approach may help us in exploiting the possible local convex property of the objective function in this class of problems, while globally these problems are not convex.
– Some optimization problems are given through a black-box oracle (see e.g., [1,9,17]). Hence, both the smoothness level of the objective function and its convex property are unknown. A unified algorithm for both convex and nonconvex optimization and for handling the smoothness level in the objective function automatically could achieve better convergence performance for this class of problems.

Our contribution in this paper mainly consists of the following aspects. First, we generalize Nesterov's AG method and present a unified problem parameter free accelerated gradient (UPFAG) method for a broad class of problems with $\nu \in [0, 1]$ in (1.2). The basic idea of this method is to combine a gradient descent step with Nesterov's AG method and maintain the monotonicity of the objective function value at the iterates generated by the algorithm. Hence, the UPFAG method contains the gradient projection and Nesterov's AG methods as special cases (see the discussions after presenting Algorithm 1). It should also be pointed out that two different line search procedures are implemented at each iteration of the UPFAG method which remove the necessity of knowing problem parameters in advance. We then show that this method achieves the best known iteration complexity

$$\mathcal{O}\left(\frac{H^{\frac{1}{\nu}}}{\epsilon^{\frac{1+\nu}{2\nu}}}\right) \tag{1.3}$$

to find at least one k such that $\|g_{X,k}\|^2 \le \epsilon$, and exhibits, similar to those in [15,25,28], the optimal complexity

$$\mathcal{O}\left(\left[\frac{H}{\epsilon}\right]^{\frac{2}{1+3\nu}}\right) \tag{1.4}$$

to find a solution $\bar{x} \in X$ such that $\Psi(\bar{x}) - \Psi^* \le \epsilon$, if f turns out to be convex.

To the best of our knowledge, this is the first time that a uniform first-order algorithm, which does not require any problem parameter information but only takes the target accuracy and a few user-defined line search parameters as the input, has been presented for solving smooth, nonsmooth, weakly smooth convex and nonconvex optimization problems. Moreover, this algorithm can also exploit a more efficient Quasi-Newton step rather than the gradient descent step for achieving the same iteration complexity bounds.

Second, by incorporating a gradient descent step into the framework of the bundle-level type methods, namely, the accelerated prox-level (APL) method presented in [15], we propose a unified APL (UAPL) method for solving a class of nonlinear programming defined in (1.1), where f satisfies (1.2). We show that this method achieves the complexity bounds in (1.4) and (1.3) for both convex and nonconvex optimization. Moreover, we simplify this method and present its fast variant, by incorporating a gradient descent step into the framework of the fast APL method [7], for solving ball-constrained and unconstrained problems. To the best of our knowledge, this is the first time that the bundle-level type methods are generalized for these nonconvex nonlinear programming problems. Indeed, the new algorithmic framework presented in this paper can be used to generalize any optimal first-order algorithm for convex optimization to solve general nonlinear (not necessarily convex) optimization problems.

The rest of the paper is organized as follows. In Sect. 2, we present the UPFAG method for solving a broad class of nonlinear programming problems where the objective function is the summation of a weakly smooth function with Hölder continuous gradient and a simple convex function, and establish the convergence results. In Sect. 3, we provide different variants of the bundle-level type methods for solving the aforementioned class of nonlinear programming. In Sect. 4, we show some numerical illustration of implementing the above-mentioned algorithms.

Notation For a differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, $h'(x)$ is the gradient of h at x. More generally, when h is a proper convex function, $\partial h(x)$ denotes the subdifferential set of h at x. For $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, $\langle x, y\rangle$ is the standard inner product in $\mathbb{R}^n$. The norm $\|\cdot\|$ is the Euclidean norm given by $\|x\| = \sqrt{\langle x, x\rangle}$, while $\|x\|_G = \sqrt{\langle Gx, x\rangle}$ for a positive definite matrix G. Moreover, we let $\mathcal{B}(\bar{x}, r)$ to be the ball with radius r centered at $\bar{x}$ i.e., $\mathcal{B}(\bar{x}, r) = \{x \in \mathbb{R}^n \mid \|x - \bar{x}\| \le r\}$. We denote I as the identity matrix. For any real number r, $\lceil r\rceil$ and $\lfloor r\rfloor$ denote the nearest integer to r from above and below, respectively. $\mathbb{R}_+$ denotes the set of nonnegative real numbers.

2 Unified Problem Parameter Free Accelerated Gradient Method

Our goal in this section is to present a unified gradient type method to solve problem (1.1) when f is smooth, weakly smooth or nonsmooth. In particular, we consider the class of problems in (1.1) such that the gradient of f is Hölder continuous in the sense of (1.2), which also implies

$$|f(y) - f(x) - \langle f'(x), y - x\rangle| \le \frac{H}{1+\nu}\|y - x\|^{1+\nu} \quad \text{for any } x, y \in X. \tag{2.1}$$

Our algorithm is stated below as Algorithm 1 which involves two line search procedures.

Algorithm 1 The unified problem-parameter free accelerated gradient (UPFAG) algorithm

```
Input: $x_0 \in X$, line search parameters $\gamma_1, \gamma_2, \gamma_3 \in (0, 1)$, and accuracy parameter $\delta > 0$.
0. Set the initial point $x_0^{ag} = x_0$, $\Lambda_0 = 0$, and $k = 1$.
1. Choose initial stepsize $\hat{\lambda}_k > 0$ and find the smallest integer $\tau_{1,k} \geq 0$ such that with
      $\eta_k = \hat{\lambda}_k \gamma_1^{\tau_{1,k}}$ and $\lambda_k = (\eta_k + \sqrt{\eta_k^2 + 4\eta_k \Lambda_{k-1}})/2$,   (2.2)
   the solutions obtained by
      $x_k^{md} = (1 - \alpha_k) x_{k-1}^{ag} + \alpha_k x_{k-1}$,   (2.3)
      $x_k = \arg\min_{u \in X} \left\{ \langle f'(x_k^{md}), u \rangle + \frac{1}{2\lambda_k} \|u - x_{k-1}\|^2 + \mathcal{X}(u) \right\}$,   (2.4)
      $\tilde{x}_k^{ag} = (1 - \alpha_k) x_{k-1}^{ag} + \alpha_k x_k$,   (2.5)
   satisfy
      $f(\tilde{x}_k^{ag}) \leq f(x_k^{md}) + \alpha_k \langle f'(x_k^{md}), x_k - x_{k-1} \rangle + \frac{\alpha_k}{2\lambda_k} \|x_k - x_{k-1}\|^2 + \delta \alpha_k$,   (2.6)
   where
      $\alpha_k = \lambda_k / \Lambda_k$ and $\Lambda_k = \sum_{i=1}^{k} \lambda_i$.   (2.7)
2. Choose initial stepsize $\hat{\beta}_k > 0$ and find the smallest integer $\tau_{2,k} \geq 0$ such that with
      $\beta_k = \hat{\beta}_k \gamma_2^{\tau_{2,k}}$,   (2.8)
   we have
      $\Psi(\bar{x}_k^{ag}) \leq \Psi(x_{k-1}^{ag}) - \frac{\gamma_3}{2\beta_k} \|\bar{x}_k^{ag} - x_{k-1}^{ag}\|^2 + \frac{1}{k}$,   (2.9)
   where
      $\bar{x}_k^{ag} = \arg\min_{u \in X} \left\{ \langle f'(x_{k-1}^{ag}), u \rangle + \frac{1}{2\beta_k} \|u - x_{k-1}^{ag}\|^2 + \mathcal{X}(u) \right\}$.   (2.10)
3. Choose $x_k^{ag}$ such that
      $\Psi(x_k^{ag}) = \min\{\Psi(x_{k-1}^{ag}), \Psi(\bar{x}_k^{ag}), \Psi(\tilde{x}_k^{ag})\}$.   (2.11)
4. Set $k \leftarrow k + 1$ and go to step 1.
```

We have the following remarks for this algorithm. First, observe that by just considering (2.3), (2.4), (2.5) and setting $x_k^{ag} = \tilde{x}_k^{ag}$, Algorithm 1 would reduce to a variant of the well-known Nesterov's optimal gradient method (see, e.g., [23]). Moreover, if replacing x_{k-1}^{ag} by x_k^{md} in (2.10) and setting $x_k^{ag} = \bar{x}_k^{ag}$, then (2.3), (2.4) and (2.10) would give the accelerated gradient (AG) method proposed by Ghadimi and Lan [11]. However, when f is nonconvex, the convergence of this AG method in [11] requires the boundedness assumption on the iterates. On the other hand, by just considering (2.10) and setting $x_k^{ag} = \bar{x}_k^{ag}$, the UPFAG algorithm would be a variant of the projected gradient method [12]. Indeed, it follows from these observations that we can possibly perform the convergence analysis of the UPFAG method for convex and nonconvex optimization separately (see the discussions after Corollary 1). It should be also pointed out that these algorithms are designed to solve the class of problems in (1.1) when f is smooth and they do not necessarily use line search procedures to specify the stepsizes. Moreover, (2.11) guarantees the objective function value at the iterates x_k^{ag} generated by the UPFAG algorithm is non-increasing. Different from the previous accelerated gradient method, such a monotonicity of the objective function value is enforced by the UPFAG method to establish convergence analysis when Ψ is nonconvex, as shown in the proof of Theorem 1.b).

Second, note that in steps 1 and 2, we implement two independent line search procedures, respectively, in (2.6) and (2.9). Indeed, we start with initial choices of stepsizes $\hat{\lambda}_k$ and $\hat{\beta}_k$, and then perform Armijo type of line searches such that certain specially designed line search conditions are satisfied. We will show in Theorem 1.a) that the two line search procedures will finish in finite number of inner iterations. One simple choice of line search in practice is to set the initial stepsizes to be some Barzilai-Borwein type stepsizes such as

$$\hat{\lambda}_k = \max\left\{\frac{\langle s_{k-1}^{md}, y_{k-1}^{md}\rangle}{\langle y_{k-1}^{md}, y_{k-1}^{md}\rangle}, \sigma\right\} \text{ for } k \ge 1, \quad \text{and} \quad \hat{\beta}_k = \max\left\{\frac{\langle s_{k-1}^{ag}, y_{k-1}^{ag}\rangle}{\langle y_{k-1}^{ag}, y_{k-1}^{ag}\rangle}, \sigma\right\} \text{ for } k > 1, \tag{2.12}$$

where $s_{k-1}^{md} = x_{k-1}^{md} - \tilde{x}_{k-1}^{ag}$, $y_{k-1}^{md} = f'(x_{k-1}^{md}) - f'(\tilde{x}_{k-1}^{ag})$, $s_{k-1}^{ag} = x_{k-1}^{ag} - x_{k-2}^{ag}$ and $y_{k-1}^{ag} = f'(x_{k-1}^{ag}) - f'(x_{k-2}^{ag})$. And we can choose $\hat{\beta}_1 = 1/\hat{H}$, if an estimation of $\hat{H}$ of the Hölder continuity constant in (1.2) is known; otherwise, we can simply set $\hat{\beta}_1$ to be a positive constant (e.g., $\hat{\beta}_1 = \min\{1/\|f'(x_0)\|, 1\}$).

Third, the Euclidean metric used in the quadratic term of the gradient projection step in (2.10) can be replaced by a variable metric. In particular, $\|\cdot\|$ can be replaced by $\|\cdot\|_{G_k}$ for some positive definite matrix G_k, i.e.,

$$\bar{x}_k^{ag} = \arg\min_{u\in X}\left\{\langle f'(x_{k-1}^{ag}), u\rangle + \frac{1}{2\beta_k}\|u - x_{k-1}^{ag}\|_{G_k}^2 + \mathcal{X}(u)\right\}. \tag{2.13}$$

This modification allows us to include some curvature information of f into the matrix G_k to have better local approximation of the function Ψ at x_{k-1}^{ag}. In this case, unit initial stepsize is often preferred, that is to set $\hat{\beta}_k = 1$ for all $k \ge 1$. In practice, we can set G_k to be some Quasi-Newton matrix, e.g., the well-known BFGS or limited memory BFGS (LBFGS) matrix (see e.g., [5,26,27]). Then, the unit step $\hat{\beta}_k = 1$ can be often accepted near the solution, which implies (2.13) will be exactly a Quasi-Newton step if $\mathcal{X}(x) \equiv 0$, and hence, a fast local convergence rate could be expected in practice (see Sect. 4 for more details). For some special structured X and $\mathcal{X}$, if the inverse G_k^{-1} is known, we may have closed formula solution of the subproblem (2.13) (e.g., see [3]). However, for more general cases, we may not have closed formula for the solution of this subproblem. Then, the alternating direction method of multipliers or primal-dual type algorithms could solve (2.13) quite efficiently, since its objective function is just a composition of a simple convex function $\mathcal{X}$ and a convex quadratic function with known inverse of the Hessian. It is also worth noting that we actually do not need to solve the subproblem (2.13) exactly if no closed formula solution exists. We can see from the proof of Theorem 1 that, instead of having (2.23) implied by the exact subproblem solution, we only need to find an approximate solution $\bar{x}_k^{ag}$ of (2.13) such that

$$\langle f'(x_{k-1}^{ag}), \bar{x}_k^{ag} - x_{k-1}^{ag}\rangle + \mathcal{X}(\bar{x}_k^{ag}) \le \mathcal{X}(x_{k-1}^{ag}) - \frac{\sigma}{\beta_k}\|\bar{x}_k^{ag} - x_{k-1}^{ag}\|^2. \tag{2.14}$$

for some $\sigma \in (0, 1)$. Hence, it is quite flexible to choose proper algorithms to solve the subproblem (2.13). We may even apply different algorithms to solve (2.13) when different solution accuracy are required. And similar as showing (2.23), the solution of the subproblem (2.13) will satisfy (2.14) as long as $G_k \succeq \sigma I$ for all k. In addition, similar as the proof in Theorem 1, if the parameter γ_3 in Algorithm 1 is set such that $0 < \gamma_3 < 2\sigma$, the line search condition (2.9) will be finally satisfied as β_k goes to zero. Hence, step 2 of Algorithm 1 is also well-defined under the variable matric and inexact solutions of subproblem (2.13). In general, by different choices of the matrix G_k, many well-studied efficient methods in nonlinear programming could be incorporated into the algorithm. Furthermore, from the complexity point of view, instead of setting the initial stepsizes given in (2.12), we could also take advantage of the line search in the previous iteration and set

$$\hat{\lambda}_k = \eta_{k-1} \quad \text{and} \quad \hat{\beta}_k = \beta_{k-1}, \tag{2.15}$$

where η_{k-1} and β_{k-1} are the accepted stepsizes in the previous $k-1$-th iteration. The choice of initial stepsizes in (2.12) is more aggressive and inherits some quasi-Newton information, and hence, could perform better in practice. However, the strategies in (2.15) would have theoretical advantages in the total number of inner iterations needed in the line search (see the discussion after Corollary 1).

In the remaining part of this section, we provide convergence properties of Algorithm 1. Noticing that $\mathcal{X}$ in problem (1.1) is not necessarily differentiable and f in (1.1) may not be a convex function, we need to define a termination criterion for this algorithm when the objective function Ψ is nonconvex. In this case, we would terminate the algorithm when the norm of the generalized projected gradient defined by

$$g_{X,k} = \frac{x_{k-1}^{ag} - \bar{x}_k^{ag}}{\beta_k} \tag{2.16}$$

is sufficiently small. Note that $g_{X,k} = f'(x_{k-1}^{ag})$ when $\mathcal{X}$ vanishes and $X = \mathbb{R}^n$. Indeed the above generalized projected gradient in constrained nonsmooth optimization plays an analogous role to that of the gradient in unconstrained smooth optimization. In particular, it can be shown that if $\|g_{X,k}\| \le \epsilon$, then $\Psi'(\bar{x}_k^{ag}) \in -\mathcal{N}_X(\bar{x}_k^{ag}) + \mathcal{B}(\epsilon^\nu(\epsilon^{1-\nu} + H\beta_k^\nu))$, where $\Psi'(\bar{x}_k^{ag}) \in \partial\Psi(\bar{x}_k^{ag})$, $\mathcal{N}_X(\bar{x}_k^{ag})$ is the normal cone of X at $\bar{x}_k^{ag}$, and $\mathcal{B}(r) := \{x \in \mathbb{R}^n : \|x\| \le r\}$ (see e.g., [11]).

To establish the convergence of the UPFAG algorithm, we also need the following simple technical result (see Lemma 3 of [16] for a slightly more general result).

Lemma 1 *Let $\{\alpha_k\}$ be a sequence of real numbers such that $\alpha_1 = 1$ and $\alpha_k \in (0, 1)$ for any $k \ge 2$. If a sequence $\{\omega_k\}$ satisfies*

$$\omega_k \le (1 - \alpha_k)\omega_{k-1} + \zeta_k, \quad k = 1, 2, \ldots, \tag{2.17}$$

then for any $k \ge 1$ we have

$$\omega_k \le \Gamma_k \sum_{i=1}^{k} (\zeta_i / \Gamma_i),$$

where

$$\Gamma_k := \begin{cases} 1, & k = 1, \\ (1 - \alpha_k)\Gamma_{k-1}, & k \ge 2. \end{cases} \tag{2.18}$$

Below, we present the main convergence properties of the UPFAG algorithm.

Theorem 1 *Let $\{x_k^{ag}\}$ be the iterates generated by Algorithm 1 and Γ_k be defined in (2.18).*

a) *The line search procedures in step 1 and step 2 of the algorithm will finish in finite number of inner iterations.*
b) *Suppose that f is bounded below over X, i.e., Ψ^* is finite. Then, for any $N \geq 1$, we have*

$$\min_{k=1,\ldots,N} \|g_{X,k}\|^2 \leq \frac{2\left[\Psi(x_0) - \Psi^* + \sum_{k=\lfloor N/2\rfloor+1}^{N} k^{-1}\right]}{\gamma_3 \sum_{k=\lfloor N/2\rfloor+1}^{N} \beta_k}, \tag{2.19}$$

where $g_{X,k}$ is defined in (2.16).
c) *Suppose that Ψ is convex and an optimal solution x^* exists for problem (1.1). Then for any $N \geq 1$, we have*

$$\Psi(x_N^{ag}) - \Psi(x^*) \leq \frac{\Gamma_N \|x_0 - x^*\|^2}{2\lambda_1} + \delta. \tag{2.20}$$

Proof We first show part a). By (2.1), we have

$$f(\bar{x}_k^{ag}) \leq f(x_{k-1}^{ag}) + \langle f'(x_{k-1}^{ag}), \bar{x}_k^{ag} - x_{k-1}^{ag}\rangle + \frac{H}{1+\nu}\|\bar{x}_k^{ag} - x_{k-1}^{ag}\|^{1+\nu}. \tag{2.21}$$

Noting that the objective in (2.10) is strongly convex and hence has a quadratic growth property near its minimum, for any $x \in X$, we have

$$\begin{aligned} &\langle f'(x_{k-1}^{ag}), \bar{x}_k^{ag} - x\rangle + \mathcal{X}(\bar{x}_k^{ag}) \leq \mathcal{X}(x) \\ &+ \frac{1}{2\beta_k}\left[\|x_{k-1}^{ag} - x\|^2 - \|\bar{x}_k^{ag} - x\|^2 - \|x_{k-1}^{ag} - \bar{x}_k^{ag}\|^2\right]. \end{aligned} \tag{2.22}$$

Letting $x = x_{k-1}^{ag}$ in the above inequality, we have

$$\langle f'(x_{k-1}^{ag}), \bar{x}_k^{ag} - x_{k-1}^{ag}\rangle + \mathcal{X}(\bar{x}_k^{ag}) \leq \mathcal{X}(x_{k-1}^{ag}) - \frac{1}{\beta_k}\|x_k^{ag} - \bar{x}_{k-1}^{ag}\|^2. \tag{2.23}$$

Summing (2.23) with (2.21), we have

$$\Psi(\bar{x}_k^{ag}) \leq \Psi(x_{k-1}^{ag}) - \left(\frac{\|\bar{x}_k^{ag} - x_{k-1}^{ag}\|^2}{\beta_k} - \frac{H\|\bar{x}_k^{ag} - x_{k-1}^{ag}\|^{1+\nu}}{1+\nu}\right). \tag{2.24}$$

Now, for $\nu \in [0,1)$, it follows from the inequality $ab \leq a^p/p + b^q/q$ with $p = \frac{2}{1+\nu}$, $q = \frac{2}{1-\nu}$, and

$$a = \frac{H}{1+\nu}\left[\frac{(1-\nu)k}{2}\right]^{\frac{1-\nu}{2}} \|x_{k-1}^{ag} - \bar{x}_k^{ag}\|^{1+\nu} \quad \text{and} \quad b = \left[\frac{2}{(1-\nu)k}\right]^{\frac{1-\nu}{2}}$$

that

$$\frac{H}{1+\nu}\|x_{k-1}^{ag} - \bar{x}_k^{ag}\|^{1+\nu} = ab \leq L(\nu, H)k^{\frac{1-\nu}{1+\nu}}\|x_{k-1}^{ag} - \bar{x}_k^{ag}\|^2 + \frac{1}{k}, \tag{2.25}$$

where

$$L(\nu, H) = \left\{\frac{H}{2\left[\frac{1+\nu}{1-\nu}\right]^{\frac{1-\nu}{2}}}\right\}^{\frac{2}{1+\nu}}. \tag{2.26}$$

Let us define

$$L(1,H)=\lim_{\nu\to 1}L(\nu,H)=\frac{H}{2}. \tag{2.27}$$

Then, (2.25) holds for all $\nu\in[0,1]$. Combining (2.24) and (2.25), we have

$$\Psi(\bar{x}_k^{ag})\le\Psi(x_{k-1}^{ag})-\frac{1-L(\nu,H)k^{\frac{1-\nu}{1+\nu}}\beta_k}{\beta_k}\|\bar{x}_k^{ag}-x_{k-1}^{ag}\|^2+\frac{1}{k}. \tag{2.28}$$

Also, by (2.1), (2.3), and (2.5), we have

$$\begin{aligned}
f(\tilde{x}_k^{ag}) &\le f(x_k^{md})+\langle f'(x_k^{md}),\tilde{x}_k^{ag}-x_k^{md}\rangle+\frac{H}{1+\nu}\|\tilde{x}_k^{ag}-x_k^{md}\|^{1+\nu}\\
&= f(x_k^{md})+\alpha_k\langle f'(x_k^{md}),x_k-x_{k-1}\rangle+\frac{H\alpha_k^{1+\nu}}{1+\nu}\|x_k-x_{k-1}\|^{1+\nu}\\
&= f(x_k^{md})+\alpha_k\langle f'(x_k^{md}),x_k-x_{k-1}\rangle\\
&\quad -\alpha_k\left(\frac{\|x_k-x_{k-1}\|^2}{2\lambda_k}-\frac{H\alpha_k^{\nu}}{1+\nu}\|x_k-x_{k-1}\|^{1+\nu}\right)+\frac{\alpha_k}{2\lambda_k}\|x_k-x_{k-1}\|^2\\
&\le f(x_k^{md})+\alpha_k\langle f'(x_k^{md}),x_k-x_{k-1}\rangle\\
&\quad -\frac{\alpha_k\left(1-2L(\nu,H)\alpha_k^{\frac{2\nu}{1+\nu}}\lambda_k\delta^{\frac{\nu-1}{1+\nu}}\right)\|x_k-x_{k-1}\|^2}{2\lambda_k}+\frac{\alpha_k}{2\lambda_k}\|x_k-x_{k-1}\|^2+\delta\alpha_k,
\end{aligned} \tag{2.29}$$

where the last inequality is obtained similar to (2.25) and $L(\nu,H)$ is defined in (2.26) and (2.27).

Now, observe that if

$$\alpha_k^{\frac{2\nu}{1+\nu}}\lambda_k\le\frac{\delta^{\frac{1-\nu}{1+\nu}}}{2L(\nu,H)}\quad\text{and}\quad\beta_k\le\frac{(2-\gamma_3)k^{\frac{\nu-1}{1+\nu}}}{2L(\nu,H)}, \tag{2.30}$$

then (2.28) and (2.29), respectively, imply (2.9) and (2.6). By (2.2) and our setting of $\alpha_k=\lambda_k/\Lambda_k=\lambda_k/(\lambda_k+\Lambda_{k-1})$, we have $\alpha_k\lambda_k=\eta_k$. Hence,

$$\alpha_k^{\frac{2\nu}{1+\nu}}\lambda_k=(\alpha_k\lambda_k)^{\frac{2\nu}{1+\nu}}\lambda_k^{\frac{1-\nu}{1+\nu}}=\eta_k^{\frac{2\nu}{1+\nu}}\lambda_k^{\frac{1-\nu}{1+\nu}}. \tag{2.31}$$

By (2.2), (2.8) and $\gamma_1,\gamma_2\in(0,1)$, we have

$$\lim_{\tau_{1,k}\to\infty}\eta_k=0,\quad\lim_{\eta_k\to 0}\lambda_k=0\quad\text{and}\quad\lim_{\tau_{2,k}\to\infty}\beta_k=0,$$

for any fixed k, which together with (2.31) imply that (2.30) will be finally satisfied in the line search procedure and therefore, (2.9) and (2.6) will essentially be satisfied. So the line search procedures in step 1 and step 2 of Algorithm 1 are well-defined and finite.

We now show part b). Noting (2.9), (2.11), and in view of (2.16), we have

$$\Psi(x_k^{ag})\le\Psi(\bar{x}_k^{ag})\le\Psi(x_{k-1}^{ag})-\frac{\gamma_3\|\bar{x}_k^{ag}-x_{k-1}^{ag}\|^2}{2\beta_k}+\frac{1}{k}=\Psi(x_{k-1}^{ag})-\frac{\gamma_3\beta_k}{2}\|g_{X,k}\|^2+\frac{1}{k}.$$

Summing up the above inequalities for k from $\lfloor N/2 \rfloor + 1$ to N and re-arranging the terms, we obtain

$$\begin{aligned}
\min_{k=1,2,\ldots,N} \|g_{X,k}\|^2 \sum_{k=\lfloor N/2\rfloor+1}^{N} \frac{\gamma_3 \beta_k}{2} &\le \min_{k=\lfloor N/2\rfloor+1,2,\ldots,N} \|g_{X,k}\|^2 \sum_{k=\lfloor N/2\rfloor+1}^{N} \frac{\gamma_3 \beta_k}{2} \\
&\le \sum_{k=\lfloor N/2\rfloor+1}^{N} \frac{\gamma_3 \beta_k}{2} \|g_{X,k}\|^2 \le \Psi(x^{ag}_{\lfloor N/2\rfloor}) - \Psi(x^{ag}_N) \\
&+ \sum_{k=\lfloor N/2\rfloor+1}^{N} \frac{1}{k} \le \Psi(x_0) - \Psi(x^*) + \sum_{k=\lfloor N/2\rfloor+1}^{N} \frac{1}{k},
\end{aligned} \tag{2.32}$$

where the last inequality follows from (2.11) and the fact that $\Psi^* \le \Psi(x^{ag}_N)$. Dividing both sides of the above inequality by $\gamma_3 \sum_{k=\lfloor N/2\rfloor+1}^{N} \frac{\beta_k}{2}$, we clearly obtain (2.19).

We now show part c). By (2.3), (2.5), (2.6), and the convexity of f, for any $x \in X$, we have

$$\begin{aligned}
f(\tilde{x}^{ag}_k) &\le f(x^{md}_k) + \alpha_k \langle f'(x^{md}_k), x_k - x_{k-1}\rangle + \frac{\alpha_k \|x_k - x_{k-1}\|^2}{2\lambda_k} + \delta\alpha_k \\
&= f(x^{md}_k) + \langle f'(x^{md}_k), \tilde{x}^{ag}_k - x^{md}_k\rangle + \frac{\alpha_k \|x_k - x_{k-1}\|^2}{2\lambda_k} + \delta\alpha_k \\
&= (1-\alpha_k)[f(x^{md}_k) + \langle f'(x^{md}_k), x^{ag}_{k-1} - x^{md}_k\rangle] + \alpha_k[f(x^{md}_k) \\
&+ \langle f'(x^{md}_k), x_k - x^{md}_k\rangle] + \frac{\alpha_k \|x_k - x_{k-1}\|^2}{2\lambda_k} + \delta\alpha_k \\
&\le (1-\alpha_k) f(x^{ag}_{k-1}) + \alpha_k f(x) + \alpha_k \langle f'(x^{md}_k), x_k - x\rangle + \frac{\alpha_k \|x_k - x_{k-1}\|^2}{2\lambda_k} + \delta\alpha_k,
\end{aligned} \tag{2.33}$$

which together with (2.11), and the convexity of $\mathcal{X}$ imply that

$$\begin{aligned}
\Psi(x^{ag}_k) &\le \Psi(\tilde{x}^{ag}_k) = f(\tilde{x}^{ag}_k) + \mathcal{X}(\tilde{x}^{ag}_k) \\
&\le (1-\alpha_k) f(x^{ag}_{k-1}) + \alpha_k f(x) + \alpha_k \langle f'(x^{md}_k), x_k - x\rangle \\
&+ \frac{\alpha_k \|x_k - x_{k-1}\|^2}{2\lambda_k} + \delta\alpha_k + (1-\alpha_k)\mathcal{X}(x^{ag}_{k-1}) + \alpha_k \mathcal{X}(x_k) \\
&= (1-\alpha_k)\Psi(x^{ag}_{k-1}) + \alpha_k f(x) + \alpha_k \langle f'(x^{md}_k), x_k - x\rangle + \frac{\alpha_k \|x_k - x_{k-1}\|^2}{2\lambda_k} + \delta\alpha_k + \alpha_k \mathcal{X}(x_k).
\end{aligned} \tag{2.34}$$

Now, by (2.4) and similar to (2.22), we have

$$\langle f'(x^{md}_k), x_k - x\rangle + \mathcal{X}(x_k) \le \mathcal{X}(x) + \frac{1}{2\lambda_k}\left[\|x_{k-1} - x\|^2 - \|x_k - x\|^2 - \|x_k - x_{k-1}\|^2\right].$$

Multiplying both sides of the above inequality by α_k and summing it up with (2.34), we obtain

$$\Psi(x^{ag}_k) \le (1-\alpha_k)\Psi(x^{ag}_{k-1}) + \alpha_k \Psi(x) + \frac{\alpha_k}{2\lambda_k}\left[\|x_{k-1} - x\|^2 - \|x_k - x\|^2\right] + \delta\alpha_k. \tag{2.35}$$

Also, note that by (2.7) and (2.18), we can easily show that

$$\Gamma_k = \frac{\lambda_1}{\sum_{i=1}^k \lambda_i} \quad \text{and} \quad \frac{\alpha_k}{\lambda_k \Gamma_k} = \frac{1}{\lambda_1} \quad \forall k \geq 1. \tag{2.36}$$

Subtracting $\Psi(x)$ from both sides of (2.35), dividing them by Γ_k, then it follows from Lemma 1 that for any $x \in X$ we have

$$\begin{aligned}\frac{\Psi(x_N^{ag}) - \Psi(x)}{\Gamma_N} &\leq \sum_{k=1}^N \frac{\alpha_k}{2\lambda_k \Gamma_k}\left[\|x_{k-1} - x\|^2 - \|x_k - x\|^2\right] + \delta \sum_{k=1}^N \frac{\alpha_k}{\Gamma_k} \\ &\leq \frac{\|x_0 - x\|^2}{2\lambda_1} + \frac{\delta}{\Gamma_N},\end{aligned} \tag{2.37}$$

where the second inequality follows from (2.36) and the fact that

$$\sum_{k=1}^N \frac{\alpha_k}{\Gamma_k} = \frac{\alpha_1}{\Gamma_1} + \sum_{k=2}^k \frac{1}{\Gamma_k}\left(1 - \frac{\Gamma_k}{\Gamma_{k-1}}\right) = \frac{1}{\Gamma_1} + \sum_{k=2}^k \left(\frac{1}{\Gamma_k} - \frac{1}{\Gamma_{k-1}}\right) = \frac{1}{\Gamma_N}. \tag{2.38}$$

Then, (2.20) follows immediately from (2.37) with $x = x^*$. □

In the next result we specify the convergence rates of the UPFAG algorithm.

Corollary 1 *Let $\{x_k^{ag}\}$ be the iterates generated by Algorithm 1. Suppose there exist some constants $\lambda > 0$ and $\beta > 0$ such that the initial stepsizes $\hat{\lambda}_k \geq \lambda$ and $\hat{\beta}_k \geq \beta$ for all $k \geq 1$.*

a) *Suppose that Ψ is bounded below over X, i.e., Ψ^* is finite. Then, for any $N \geq 1$, we have*

$$\min_{k=1,\ldots,N} \|g_{X,k}\|^2 \leq \frac{8[\Psi(x_0) - \Psi^* + 1]}{\gamma_2}\left[\frac{8L(\nu, H)}{(2-\gamma_3)N^{\frac{2\nu}{1+\nu}}} + \frac{1}{\beta N}\right]. \tag{2.39}$$

b) *Suppose that f is convex and an optimal solution x^* exists for problem (1.1). Then for any $N \geq 1$, we have*

$$\Psi(x_N^{ag}) - \Psi(x^*) \leq \frac{4\|x_0 - x^*\|^2}{\gamma_1 N^{\frac{1+3\nu}{1+\nu}}}\left[\frac{2L(\nu, H)}{\delta^{\frac{1-\nu}{1+\nu}}} + \frac{1}{\lambda}\right] + \delta. \tag{2.40}$$

Proof Since $\hat{\lambda}_k \geq \lambda$ and $\hat{\beta}_k \geq \beta$ for all $k \geq 1$, then it follows from (2.2), (2.8), (2.30) and $\eta_k = \alpha_k \lambda_k$ that

$$\beta_k \geq \gamma_2 \min\left\{\frac{(2-\gamma_3)k^{\frac{\nu-1}{1+\nu}}}{2L(\nu, H)}, \beta\right\} \quad \text{and} \quad \alpha_k^{\frac{2\nu}{1+\nu}} \lambda_k \geq \gamma_1 \min\left\{\frac{\delta^{\frac{1-\nu}{1+\nu}}}{2L(\nu, H)}, \lambda \alpha_k^{\frac{\nu-1}{1+\nu}}\right\}, \tag{2.41}$$

which together with

$$\begin{aligned}\sum_{k=\lfloor N/2 \rfloor+1}^N k^{\frac{1-\nu}{1+\nu}} &\leq \int_{x=\lfloor N/2 \rfloor+1}^{N+1} x^{\frac{1-\nu}{1+\nu}} dx \leq 4N^{\frac{2}{1+\nu}}, \\ \sum_{k=\lfloor N/2 \rfloor+1}^N \frac{1}{k} &\leq \int_{x=\lfloor N/2 \rfloor}^N \frac{dx}{x} = \ln \frac{N}{\lfloor N/2 \rfloor} \leq 1,\end{aligned} \tag{2.42}$$

and arithmetic-harmonic mean inequality imply that

$$\begin{aligned}\sum_{k=\lfloor N/2\rfloor+1}^{N}\beta_k &\geq \sum_{k=\lfloor N/2\rfloor+1}^{N}\gamma_2\min\left\{\frac{(2-\gamma_3)k^{\frac{\nu-1}{1+\nu}}}{2L(\nu,H)},\beta\right\}\\ &\geq \frac{\gamma_2N^2}{4\sum_{k=\lfloor N/2\rfloor+1}^{N}\max\left\{\frac{2L(\nu,H)}{(2-\gamma_3)k^{\frac{\nu-1}{1+\nu}}},\frac{1}{\beta}\right\}}\\ &\geq \frac{\gamma_2N^2}{4\sum_{k=\lfloor N/2\rfloor+1}^{N}\left\{2(2-\gamma_3)^{-1}L(\nu,H)k^{\frac{1-\nu}{1+\nu}}+\beta^{-1}\right\}}\\ &\geq \frac{\gamma_2}{4\left(8(2-\gamma_3)^{-1}L(\nu,H)N^{\frac{-2\nu}{1+\nu}}+(\beta N)^{-1}\right)}. \end{aligned} \tag{2.43}$$

Combining the above relation with (2.19), we clearly obtain (2.39).

Now, observing (2.41) and the facts that $\alpha_k \in (0,1]$ and $\nu \in (0,1]$, we have

$$\alpha_k^{\frac{2\nu}{1+\nu}}\lambda_k \geq \gamma_1\min\left\{\frac{\delta^{\frac{1-\nu}{1+\nu}}}{2L(\nu,H)},\lambda\right\},$$

which together with (2.36) imply that

$$\lambda_k \geq \left(\frac{\lambda_1}{\Gamma_k}\right)^{\frac{2\nu}{1+3\nu}}\gamma_1^{\frac{1+\nu}{1+3\nu}}\min\left\{\left(\frac{\delta^{\frac{1-\nu}{1+\nu}}}{2L(\nu,H)}\right)^{\frac{1+\nu}{1+3\nu}},\lambda^{\frac{1+\nu}{1+3\nu}}\right\}.$$

Noticing this observation, defining $c=\frac{1+\nu}{1+3\nu}$, then by (2.18) and (2.36), we have

$$\frac{1}{\Gamma_k^c}-\frac{1}{\Gamma_{k-1}^c}=\frac{1}{\Gamma_k^c}-\frac{(1-\alpha_k)^c}{\Gamma_k^c}\geq\frac{c\alpha_k}{\Gamma_k^c}=\frac{c\lambda_k\Gamma_k^{1-c}}{\lambda_1}\geq\frac{c\gamma_1^c}{\lambda_1^c}\min\left\{\frac{\delta^{\frac{1-\nu}{1+3\nu}}}{[2L(\nu,H)]^c},\lambda^c\right\},$$

where the first inequality follows from the fact that $1-(1-\alpha)^c \geq c\alpha$ for all $\alpha \in (0,1]$ and $c \in [1/2,1]$. By the above inequality and noticing $\Gamma_0 = 1$, similar to (2.43), for any $N \geq 1$ we obtain

$$\frac{1}{\Gamma_N^c}\geq\frac{c\gamma_1^cN^2}{\lambda_1^c\sum_{k=1}^{N}\left\{[2L(\nu,H)\delta^{\frac{\nu-1}{1+\nu}}]^c+\lambda^{-c}\right\}}\geq\frac{c\gamma_1^cN}{\lambda_1^c\left\{[2L(\nu,H)\delta^{\frac{\nu-1}{1+\nu}}]^c+\lambda^{-c}\right\}},$$

which together with the facts that $c^{\frac{1}{c}} \geq 1/4$ and $(a+b)^{\frac{1}{c}} \leq 2(a^{\frac{1}{c}}+b^{\frac{1}{c}})$ for any $a, b > 0$, imply that

$$\Gamma_N \leq \frac{8\lambda_1}{\gamma_1N^{\frac{1+3\nu}{1+\nu}}}\left[\frac{2L(\nu,H)}{\delta^{\frac{1-\nu}{1+\nu}}}+\frac{1}{\lambda}\right].$$

Combining the above relation with (2.20), clearly we obtain (2.40). □

We now add a few remarks about the results obtained in Corollary 1. First, for simplicity let us assume $\beta \geq (2-\gamma_3)/L(\nu,H)$. Then, by (2.39), we conclude that the number of iterations performed by the UPFAG method to have $\|g_{X,k}\|^2 \le \epsilon$ for at least one k, after disregarding some constants, is bounded by[1]

$$\mathcal{O}\left(H^{\frac{1}{\nu}}\left[\frac{\Psi(x_0)-\Psi^*}{\epsilon}\right]^{\frac{1+\nu}{2\nu}}\right). \tag{2.44}$$

This bound can be also obtained by using a variable metric in (2.13) instead of the Euclidean one in (2.10). When G_k is uniformly bounded from above, i.e., $G_k \preceq MI$ for some constant $M > 0$, the projected gradients defined according to both (2.10) and (2.13) are equivalent. Moreover, when $\nu = 1$, the above bound will reduce to the best known iteration complexity for the class of nonconvex functions with Lipschitz continuous gradient which is also known for the steepest descent method for unconstrained problems [23], and the projected gradient method for composite problems in Ghadimi, Lan and Zhang [12]. In this case, a similar bound is also obtained by the AG method [10], which however, for composite problems, relies on an additional assumption that the iterates are bounded. In spite of this similarity in iteration complexity of these methods, the UPFAG method can outperform the other two methods in practice (see Sect. 4 for more details). On the other hand, one possible advantage of this AG method in [10] exists in that it can separate the affects of the Lipschitz constants of smooth convex terms.

Second, by choosing $\delta = \epsilon/2$ and assuming that $\lambda \ge \delta^{\frac{1-\nu}{1+\nu}}/L(\nu, H)$, (2.40) implies that the UPFAG method can find a solution $\bar{x}$ such that $\Psi(\bar{x}) - \Psi(x^*) \le \epsilon$, after disregarding some constants, in at most

$$\mathcal{O}\left(\left[\frac{H\|x_0 - x^*\|^{1+\nu}}{\epsilon}\right]^{\frac{2}{1+3\nu}}\right) \tag{2.45}$$

number of iterations which is optimal for convex programming [21]. Note that (2.45) is in the same order of magnitude as the bound obtained by the universal fast gradient method proposed by Nesterov [25] for convex optimization problems. While this method also uses line search procedure to specify stepsizes, it does not applicable to nonconvex optimization. On the other hand, the UPFAG method does not need to know the convexity of the objective function as a prior knowledge. It treats both the convex and nonconvex optimization problems in a unified way. In any case, this method always achieves the complexity bound in (2.44) and when the objective function happens to be convex, it would also achieve the optimal complexity bound in (2.45). This unified approach, as mentioned before, can help us to exploit local convexity structure of general nonconvex problems and find a stationary point faster.

Finally, it is interesting to find the number of gradient computations at each iteration of Algorithm 1 assuming the initial stepsizes $\hat{\lambda}_k \ge \lambda$ and $\hat{\beta}_k \ge \beta$ for all $k \ge 1$. According to (2.8) and (2.41), we conclude that, after disregarding some constants, $\tau_{2,k} \le \log k$ which implies that the number of gradient computations at point x_{k-1}^{ag} is bounded by $\log k$. Similarly, we obtain that the number of gradient computations at point x_k^{md} is also bounded by $\log k$. Hence, the total number of gradient computations at the k-th iteration is bounded by $2\log k$. On the other hand, suppose that we choose $\hat{\beta}_k$ and $\hat{\lambda}_k$ according to (2.15). Then, we have

$$\tau_{1,k} = (\log \eta_k - \log \eta_{k-1})/\log \gamma_1 \quad \text{and} \quad \tau_{2,k} = (\log \beta_k - \log \beta_{k-1})/\log \gamma_2.$$

[1] This complexity bound was also derived for the gradient descent method as a homework assignment given by the second author in Spring 2014, later summarized by one of the class participants in [30]. However this development requires the problem to be unconstrained and the parameters H and ν be a given priori.

So the number of gradient evaluations in step 1 and step 2 at the k-th iteration is bounded by

$$1 + \tau_{1,k} = 1 + (\log \eta_k - \log \eta_{k-1})/\log \gamma_1 \quad \text{and} \quad 1 + \tau_{2,k} = 1 + (\log \beta_k - \log \beta_{k-1})/\log \gamma_2,$$

which implies the total number of gradient evaluations in step 1 and step 2 is bounded by

$$N_\eta = N + \sum_{k=1}^{N} \tau_{1,k} = N + \frac{\log \eta_N - \log \eta_0}{\log \gamma_1} \quad \text{and}$$
$$N_\beta = N + \sum_{k=1}^{N} \tau_{2,k} = N + \frac{\log \beta_N - \log \beta_0}{\log \gamma_2}.$$

Note that (2.41) implies $N_\eta \le N + c_1$ and $N_\beta \le N + c_2 \log N$ for some positive constants c_1 and c_2. Hence, the above relations show that the average number of gradient computations at each iteration is bounded by a constant, which is less than the aforementioned logarithmic bound $\log k$ obtained for the situations where $\hat{\beta}_k$ and $\hat{\lambda}_k$ are chosen according to (2.2) and (2.8). However, in (2.2) and (2.8) the algorithm allows more freedom to choose initial stepsizes.

One possible drawback of the UPFAG method is that we need to fix the accuracy δ before running the algorithm and if we want to change it, we should run the algorithm from the beginning. Moreover, we need to implement two line search procedures to find β_k and λ_k in each iteration. In the next section, we address these issues by presenting some problem-parameter free bundle-level type algorithms which do not require a fixed target accuracy in advance and only performs one line search procedure in each iteration. These bundle-level type methods can sometimes outperform gradient type methods in practice (see Sect. 4 for more details).

3 Unified Bundle-Level Type Methods

Our goal in this section is to generalize bundle-level type methods, originally designed for convex programming, for solving a class of possibly nonconvex nonlinear programming problems. Specifically, in Sect. 3.1, we introduce a unified bundle-level type method by incorporating a local search step into an accelerated prox-level method and establish its convergence properties under the boundedness assumption of the feasible set. In Sect. 3.2, we simplify this algorithm and provide its fast variants for solving ball-constrained problems and unconstrained problems with bounded level sets. To the best of our knowledge, this is the first time that bundle-level type methods are proposed for solving a class of nonconvex optimization problems.

3.1 Unified Accelerated Prox-level Method

In this subsection, we generalize the accelerate prox-level (APL) method presented by Lan [15] to solve a class of nonlinear programming given in (1.1), where f has Hölder continuous gradient on X. Lan [15] showed that the APL method is uniformly optimal for solving problem (1.1) when f is convex and satisfies (1.2). Here, we combine the framework of this algorithm with a gradient descent step and present a unified accelerated prox-level (UAPL) method for solving both convex and nonconvex optimization.

As the bundle-level type methods, we introduce some basic definitions about the objective function and the feasible set. We first define a function $h(y, x)$ for a given $y \in X$, as

$$h(y, x) = f(y) + \langle f'(y), x - y\rangle + \mathcal{X}(x) \quad \text{for any } x \in X. \tag{2.1}$$

Note that if f is convex, then we have $h(y, x) \le f(x) + \mathcal{X}(x) = \Psi(x)$ and hence $h(y, x)$ defines a lower bound for $\Psi(x)$. Also, let $\mathcal{S}_\Psi(l)$ be the level set of Ψ given by $\mathcal{S}_\Psi(l) = \{x \in X : \Psi(x) \le l\}$ and define a convex compact set X' as a localizer of the level set $\mathcal{S}_\Psi(l)$ if it satisfies $\mathcal{S}_\Psi(l) \subseteq X' \subseteq X$. Then, it can be shown [16] that, when Ψ is convex, $\min\{l, \underline{h}(y)\} \le \Psi(x)$ for any $x \in X$, where

$$\underline{h}(y) = \min\{h(y, x) : x \in X'\}. \tag{2.2}$$

Using the above definitions, we present a unified accelerated prox-level (UAPL) algorithm, Algorithm 2, for nonlinear programming.

Algorithm 2 The unified accelerated prox-level (UAPL) algorithm

```
Input: $p_0 \in X$, $\alpha_t \in (0,1)$ with $\alpha_1 = 1$, and algorithmic parameter $\eta \in (0,1)$.
Set $p_1 \in \text{Argmin}_{x\in X} h(p_0, x)$, $\text{lb}_1 = h(p_0, p_1)$, $x_0^{ag} = p_0$, and $k = 0$.

For $s = 1, 2, \ldots$:
    Set $\hat{x}_0^{ag} = p_s$, $\bar{\Psi}_0 = \Psi(\hat{x}_0^{ag})$, $\underline{\Psi}_0 = \text{lb}_s$, and $l = \eta\underline{\Psi}_0 + (1-\eta)\bar{\Psi}_0$. Also, let $x_0 \in X$ and the initial
    localizer $X'_0$ be arbitrarily chosen, say $x_0 = p_s$ and $X'_0 = X$.

    For $t = 1, 2, \ldots$:
        1. Update lower bound: set $x_t^{md} = (1-\alpha_t)\hat{x}_{t-1}^{ag} + \alpha_t x_{t-1}$ and $\underline{\Psi}_t := \max\left\{\underline{\Psi}_{t-1}, \min\{l, \underline{h}_t\}\right\}$,
        where $\underline{h}_t \equiv \underline{h}(x_t^{md})$ is defined in (2.2) with $X' = X'_{t-1}$.
        2. Update the prox-center: set
            $$x_t = \text{argmin}_{x\in X'_{t-1}} \left\{\|x - x_0\|^2 : h(x_t^{md}, x) \le l\right\}. \tag{3.3}$$
        If (3.3) is infeasible, set $x_t = \hat{x}_{t-1}^{ag}$.
        3. Update upper bound: set $k \leftarrow k + 1$ and choose $\hat{x}_t^{ag}$ such that $\Psi(\hat{x}_t^{ag}) =$
        $\min\{\Psi(\hat{x}_{t-1}^{ag}), \Psi(\tilde{x}_t^{ag}), \Psi(\bar{x}_k^{ag})\}$, where $\tilde{x}_t^{ag} = (1-\alpha_t)\hat{x}_{t-1}^{ag} + \alpha_t x_t$ and $\bar{x}_k^{ag}$ is obtained by
        step 2 of Algorithm 1. Set $\bar{\Psi}_t = \Psi(\hat{x}_t^{ag})$ and $x_k^{ag} = \hat{x}_t^{ag}$.

        4. Termination: If $\underline{\Psi}_t \le \bar{\Psi}_t$ and $\bar{\Psi}_t - \underline{\Psi}_t \le [1 - \frac{1}{2}\min\{\eta, 1-\eta\}](\bar{\Psi}_0 - \underline{\Psi}_0)$, then terminate this
        phase (loop) and set $p_{s+1} = \hat{x}_t^{ag}$ and $\text{lb}_{s+1} = \underline{\Psi}_t$.

        5. Update localizer: choose an arbitrary $X'_t$ such that $\underline{X}_t \subseteq X'_t \subseteq \bar{X}_t$, where

            $$\underline{X}_t = \left\{x \in X'_{t-1} : h(x_t^{md}, x) \le l\right\} \text{ and } \bar{X}_t = \{x \in X : \langle x_t - x_0, x - x_t\rangle \ge 0\}. \tag{3.4}$$

    End
End
```

We now make a few remarks about the above algorithm. First, note that the updating of $\bar{x}_k^{ag}$ in step 3 of the UAPL algorithm is essentially a gradient descent step, and hence without this update, the UAPL algorithm would reduce to a simple variant of the APL method in [15] for convex programming. However, this update is required to establish convergence for the case when the objective function is nonconvex. Second, this UAPL algorithm has two nested loops. The outer loop called phase is counted by index s. The number of iterations of the inner loop in each phase is counted by index t. If we make a progress in reducing the gap between the lower and upper bounds on Ψ, we terminate the current phase (inner loop) in step 4 and go to the next one with a new lower bound. As shown in [15], the number of steps in each phase is finite when Ψ is convex. However, when Ψ is nonconvex, $\underline{\Psi}$ is not necessary a lower bound on Ψ, and hence the termination criterion in step 4 may not be satisfied. In this case, we could still provide the convergence of the algorithm in terms of the projected gradient defined in (2.16) because of the gradient descent step incorporated in step 3.

The following Lemma [15] shows some properties of the UAPL algorithm by generalizing the prox-level method in [4].

Lemma 2 *Assume that Ψ is convex and bounded below over X, i.e., Ψ^* is finite. Then, the following statements hold for each phase of Algorithm 2.*

a) *$\{X'_t\}_{t\geq 0}$ is a sequence of localizers of the level set $\mathcal{S}_\Psi(l)$.*
b) *$\underline{\Psi}_0 \leq \underline{\Psi}_1 \leq \ldots \leq \underline{\Psi}_t \leq \Psi^* \leq \bar{\Psi}_t \leq \ldots \leq \bar{\Psi}_1 \leq \bar{\Psi}_0$ for any $t \geq 1$.*
c) *$\emptyset \neq \underline{X}_t \subseteq \bar{X}_t$ for any $t \geq 1$ and hence, step 5 is always feasible unless the current phase terminates.*

Now, we can present the main convergence properties of the above UAPL algorithm.

Theorem 2 *Let the feasible set X be bounded.*

a) *Suppose Ψ is bounded below over X, i.e., Ψ^* is finite. The total number of iterations performed by the UAPL method to have $\|g_{X,k}\|^2 \leq \epsilon$ for at least one k, after disregarding some constants, is bounded by (2.44).*
b) *Suppose that f is convex and an optimal solution x^* exists for problem (1.1). Then, the number of phases performed by the UAPL method to find a solution $\bar{x}$ such that $\Psi(\bar{x}) - \Psi(x^*) \leq \epsilon$, is bounded by*

$$S(\epsilon) = \left\lceil \max\left\{0, \log_{\frac{1}{q}} \frac{H \max_{x,y\in X} \|x-y\|^{1+\nu}}{(1+\nu)\epsilon}\right\}\right\rceil, \tag{3.5}$$

where

$$q = 1 - \frac{1}{2}\min\{\eta, 1-\eta\}. \tag{3.6}$$

In addition, by choosing $\alpha_t = 2/(t+1)$ for any $t \geq 1$, the total number of iterations to find the aforementioned solution $\bar{x}$ is bounded by

$$S(\epsilon) + \frac{1}{1 - q^{\frac{2}{3\nu+1}}} \left(\frac{4\sqrt{3}H \max_{x,y\in X} \|x-y\|^{1+\nu}}{\eta\theta(1+\nu)\epsilon}\right)^{\frac{2}{3\nu+1}}. \tag{3.7}$$

Proof First, note that part a) can be established by essentially following the same arguments as those in Theorem 1.b) and Corollary 1.a) due to step 3 of Algorithm 2. Second, due to the termination criterion in step 4 of Algorithm 2, for any phase $s \geq 1$, we have

$$\Psi(p_{s+1}) - \text{lb}_{s+1} \leq q[\Psi(p_s) - \text{lb}_s],$$

which by induction and together with the facts that $\text{lb}_1 = h(p_0, p_1)$ and $\text{lb}_s \leq \Psi(x^*)$, clearly imply

$$\Psi(p_s) - \Psi(x^*) \leq [\Psi(p_1) - h(p_0, p_1)]\, q^{s-1}. \tag{3.8}$$

This relation, as shown in Theorem 4 of [15] for convex programming, implies (3.5). Third, (3.7) is followed by (3.5) and [15, Proposition 2, Theorem 3]. □

We now add a few remarks about the above results. First, note that the UAPL and UPFAG methods essentially have the same mechanism to ensure the global convergence when the problem (1.1) is nonconvex. To the best of our knowledge, this is the first time that a bundle-level type method is proposed for solving a class of possibly nonconvex nonlinear programming problems. It should be also mentioned that similar complexity results can be obtained by using (2.13) instead of (2.10) in step 3 of the UAPL method. Second, note that the bound in (3.7) is in the same order of magnitude as the optimal bound in (2.45) for convex programming. However, to obtain this bound, we need the boundedness assumption on the feasible set X, although we do not need the target accuracy as a priori. Third, parts a) and c) of Theorem 2 imply that the UAPL method can uniformly solve weakly smooth nonconvex and convex problems without requiring any problem parameters. In particular, it achieves the best known convergence rate for nonconvex problems and its convergence rate is optimal if the problem turns out to be convex.

Finally, in steps 1 and 2 of the UAPL algorithm, we need to solve two subproblems which can be time consuming. Moreover, to establish the convergence of this algorithm, we need the boundedness assumption on X as mentioned above. In the next subsection, we address these issues by exploiting the framework of another bundle-level type method which can significantly reduce the iteration cost.

3.2 Unified Fast Accelerated Prox-level Method

In this subsection, we aim to simplify the UAPL method for solving ball-constrained problems and unconstrained problems with bounded level sets. Recently, Chen et al. [7] presented a simplified variant of the APL method, namely fast APL (FAPL) method, for ball constrained convex problems. They showed that the number of subproblems in each iteration is reduced from two to one and presented an exact method to solve the subproblem.

In this subsection, we first generalize the FAPL method for ball-constrained nonconvex problems and then discuss how to extend it for unconstrained problems with bounded level sets. It should be mentioned that throughout this subsection, we assume that the simple nonsmooth convex term vanishes in the objective function i.e., $\mathcal{X} \equiv 0$ in (1.1). Below, we present the unified FAPL (UFAPL) method to sovle the problem (1.1) with the ball constraint, i.e., $X = \mathcal{B}(\bar{x}, R)$.

We now add a few remarks about this algorithm. First, note that we do not need to solve the subproblem (2.2) in the UFAPL method. Moreover, the subproblem (3.3) in the UFAPL method is to project $\bar{x}$ over a closed polyhedral. There exist quite efficient methods for performing such a projection on a polyhedral (see e.g., [7,13]). When (2.13) is used and $G_k = I$, the subproblem associated with finding $\bar{x}_k^{ag}$ in step 3 of the UFAPL method has a closed-form solution. Hence, there is only one subproblem to be solved in each iteration of the UFAPL method and this subproblem can be solved quite efficiently. In addition, since the objective function may be nonconvex, when a phase s starts in the UAPL or the UFAPL, the lower bound $\underline{\Psi}_0$ could be even larger than the upper bound $\bar{\Psi}_0$. When this happens, from practical efficiency point of view, one may simply restart the algorithm at the current iteration point p_s or may even only perform the local nonlinear search iterations starting at p_s.

Second, note that the UFAPL algorithm can be terminated in step 3 or step 4. Moreover, by combining the convergence results in [7] and applying similar techniques used in showing the Theorem 1, we can establish complexity results similar to the Theorem 2 for the UFAPL method. For simplicity, we do not repeat these arguments here.

Finally, we can extend the above results for the UFAPL method to unconstrained problems. In particular, suppose the level set

$$\mathcal{S}_0 = \{x \in \mathbb{R}^n \ |\Psi(x) \le \Psi(x_0)\},$$

is bounded, where x_0 is the initial point for the UFAPL method. Now, consider the ball-constrained problem

$$\min_{x \in \mathcal{B}(x_0, R)} \Psi(x), \tag{3.9}$$

such that $R = \max_{x,y \in \mathcal{S}_0} \|x - y\| + \delta$ for a given $\delta > 0$. To solve this problem, we could apply the UFAPL method with small modifications. Specifically, we use $X = \mathbb{R}^n$ to find $\bar{x}_k^{ag}$ in step 3 of this method. Now, let $\{x_k^{ag}\}_{k \ge 1}$ be generated by this modified UFAPL method. By steps 3 and 4 of this method, we clearly have $\Psi(x_k^{ag}) \le \Psi(x_{k-1}^{ag})$ for all $k \ge 1$, which implies that $x_k^{ag} \in \mathcal{S}_0$ for all $k \ge 1$. Hence, all generated points $\{x_k^{ag}\}_{k \ge 1}$ lie in the interior of the aforementioned ball $\mathcal{B}(x_0, R)$ due to $\delta > 0$, which consequently implies that the optimal solution of the problem (3.9) is the same as the that of the unconstrained problem $\min_{x \in \mathbb{R}^n} \Psi(x)$. Therefore, under the boundedness assumption on $\mathcal{S}_0$, we can apply the above modified UFAPL method to solve the ball-constrained problem (3.9) in order to solve the original unconstrained problem.

Algorithm 3 The unified fast accelerated prox-level (UFAPL) algorithm

```
Input: $p_0 \in \mathcal{B}(\bar{x}, R)$ and $\eta, \theta \in (0, 1)$.
Set $p_1 \in \text{Argmin}_{x \in \mathcal{B}(\bar{x}, R)} h(p_0, x)$, $\text{lb}_1 = h(p_0, p_1)$, $x_0^{ag} = p_0$, and $k = 0$.

For $s = 1, 2, \ldots$:
    Set $\hat{x}_0^{ag} = p_s$, $\bar{\Psi}_0 = \Psi(\hat{x}_0^{ag})$, $\underline{\Psi}_0 = \text{lb}_s$, $l = \eta\underline{\Psi}_0 + (1 - \eta)\bar{\Psi}_0$, and $X_0' = X = \mathbb{R}^n$. Also, let
    $x_0 \in \mathcal{B}(\bar{x}, R)$ be arbitrary given.

    For $t = 1, 2, \ldots$:
        1. Set $x_t^{md} = (1 - \alpha_t)\hat{x}_{t-1}^{ag} + \alpha_t x_{t-1}$ and define $\underline{X}_t$ as in (3.4).
        2. Update the prox-center: Let $x_t$ be computed by (3.3) with $x_0 = \bar{x}$, i.e., $x_t = \text{argmin}_{x \in \underline{X}_t} \|x - \bar{x}\|^2$.
        3. Update the lower bound: set $k \leftarrow k + 1$ and choose $\hat{x}_t^{ag}$ such that $\Psi(\hat{x}_t^{ag}) =$
        $\min\{\Psi(\hat{x}_{t-1}^{ag}), \Psi(\bar{x}_k^{ag})\}$, where $\bar{x}_k^{ag}$ is obtained by step 2 of Algorithm 1 with $X = \mathcal{B}(\bar{x}, R)$.
        If $\underline{X}_t = \emptyset$ or $\|x_t - \bar{x}\| > R$, then terminate this phase with $x_k^{ag} = \hat{x}_t^{ag}$, $p_{s+1} = \hat{x}_t^{ag}$, and
        $\text{lb}_{s+1} = l$.

        4. Update upper bound: let $\tilde{x}_t^{ag} = (1 - \alpha_t)\hat{x}_{t-1}^{ag} + \alpha_t x_t$. If $\Psi(\tilde{x}_t^{ag}) < \Psi(\hat{x}_t^{ag})$, then set $x_k^{ag} =$
        $\hat{x}_t^{ag} = \tilde{x}_t^{ag}$ and $\bar{\Psi}_t = \Psi(\hat{x}_t^{ag})$. If $\bar{\Psi}_t \le l + \theta(\bar{\Psi}_0 - l)$, then terminate this phase (loop) and set
        $p_{s+1} = \hat{x}_t^{ag}$ and $\text{lb}_{s+1} = \text{lb}_s$.

        5. Update localizer: choose any polyhedral $X_t'$ such that $\underline{X}_t \subseteq X_t' \subseteq \bar{X}_t$, where $\underline{X}_t$ and $\bar{X}_t$ are
        defined in (3.4) with $X = \mathbb{R}^n$.
    End
End
```

4 Numerical Experiments

In this section, we show the performance of our algorithms for solving two support vector machine (SVM) problems. In particular, given data points $\{(u_i, v_i)\}_{i=1}^m$ we consider the SVM problem defined as

Table 1 Average required number of iterations (Iter(k)), runtime ($T(s)$), objective value, and classification error found till reaching a desired accuracy for $\|g_X(\bar{x}^*)\|^2$ over 10 random generated instances of the SVM problem with the sigmoid loss function

$\|g_X(\bar{x}^*)\|^2$		$< 10^{-2}$	$< 10^{-4}$	$< 10^{-6}$	$< 10^{-8}$	$< 10^{-10}$	$< 10^{-12}$
$m = 1000, n = 2000$							
LBFGS	Iter(k)	1	12	38	43.9	48.2	53.3
	$T(s)$	0	0	0.1	0.1	0.1	0.1
	$\Psi(x_k^{ag})$	1.3	0.289	0.277	0.277	0.277	0.277
	$er(x_k^{ag})$	49.64	32.38	32.06	32.07	32.07	32.07
UPFAG-LBFGS	Iter(k)	1	7.6	16.2	22.5	28.1	32.2
	$T(s)$	0	0	0	0	0.1	0.1
	$\Psi(x_k^{ag})$	1.3	0.295	0.277	0.277	0.277	0.277
	$er(x_k^{ag})$	49.64	32.65	32.03	32.03	32.07	32.07
UPFAG-fullBB	Iter(k)	1	8.6	20.7	32.6	43.9	55.9
	$T(s)$	0	0	0	0	0.1	0.1
	$\Psi(x_k^{ag})$	1.3	0.295	0.277	0.277	0.277	0.277
	$er(x_k^{ag})$	49.64	32.66	32.1	32.05	32.06	32.07
UPFAG-GradDescent	Iter(k)	1	36.8	101.8	227.6	422.3	649.6
	$T(s)$	0	0	0.1	0.3	0.5	0.7
	$\Psi(x_k^{ag})$	1.3	0.298	0.277	0.277	0.277	0.277
	$er(x_k^{ag})$	49.64	32.53	32.09	32.05	32.05	32.06
UFAPL	Iter(k)	1	10.2	17.9	24.9	29.2	35.3
	$T(s)$	0	0.1	0.1	0.1	0.1	0.2
	$\Psi(x_k^{ag})$	1.3	0.288	0.277	0.277	0.277	0.277
	$er(x_k^{ag})$	49.64	32.33	32.06	32.06	32.06	32.04
UAPL	Iter(k)	1	10.4	28.2	45.8	49.9	54.2
	$T(s)$	0.1	0.9	2.7	4.5	4.9	5.4
	$\Psi(x_k^{ag})$	1.59	0.287	0.277	0.277	0.277	0.277
	$er(x_k^{ag})$	38.64	32.5	32.13	32.08	32.09	32.08
$m = 4000, n = 8000$							
LBFGS	Iter(k)	1	29.8	63.6	189	303.4	390.7
	$T(s)$	0	0.1	0.2	0.5	0.7	0.9
	$\Psi(x_k^{ag})$	1.3	0.173	0.122	0.12	0.12	0.119
	$er(x_k^{ag})$	49.68	33.02	31.75	31.78	31.8	31.82
UPFAG-LBFGS	Iter(k)	1	14.3	29.4	54.1	69.4	86.5
	$T(s)$	0	0.1	0.2	0.4	0.4	0.5
	$\Psi(x_k^{ag})$	1.3	0.175	0.121	0.119	0.119	0.119
	$er(x_k^{ag})$	49.68	33.22	31.85	31.8	31.82	31.82
UPFAG-fullBB	Iter(k)	1	17.3	50.1	89.2	129.3	162
	$T(s)$	0	0.1	0.3	0.5	0.7	0.8
	$\Psi(x_k^{ag})$	1.3	0.208	0.12	0.119	0.119	0.119
	$er(x_k^{ag})$	49.68	34.63	31.92	31.83	31.87	31.86

Table 1 continued

$\|g_X(\bar{x}^*)\|^2$		$< 10^{-2}$	$< 10^{-4}$	$< 10^{-6}$	$< 10^{-8}$	$< 10^{-10}$	$< 10^{-12}$
UPFAG-GradDescent	Iter(k)	1	140.7	527.2	1289.2	>3000	–
	$T(s)$	0	0.6	2.2	5.4	–	–
	$\Psi(x_k^{ag})$	1.3	0.184	0.119	0.119	–	–
	$er(x_k^{ag})$	49.68	33.15	31.78	31.79	–	–
UFAPL	Iter(k)	1	12.5	32.5	56.1	70.4	95
	$T(s)$	0	0.1	0.3	0.5	0.6	0.9
	$\Psi(x_k^{ag})$	0.95	0.18	0.121	0.12	0.12	0.119
	$er(x_k^{ag})$	42.28	33.07	31.79	31.79	31.77	31.79
UAPL	Iter(k)	1	39.7	87	183	269.5	332.1
	$T(s)$	1.5	29.9	66.6	138.1	202.2	249.7
	$\Psi(x_k^{ag})$	0.95	0.166	0.121	0.12	0.119	0.119
	$er(x_k^{ag})$	42.28	32.88	31.8	31.77	31.78	31.78

Table 2 Required number of iterations (Iter(k)), runtime ($T(s)$), objective value, and classification error found till reaching a desired accuracy for $\|g_X(\bar{x}^*)\|^2$ over real data of the SVM problem with the sigmoid loss function

$\|g_X(\bar{x}^*)\|^2$		$< 10^{-2}$	$< 10^{-4}$	$< 10^{-6}$	$< 10^{-8}$	$< 10^{-10}$	$< 10^{-12}$
*a*6*a*							
LBFGS	Iter(k)	2	2	6	60	85	107
	$T(s)$	0.2	0.2	0.2	0.3	0.4	0.4
	$\Psi(x_k^{ag})$	0.488	0.488	0.483	0.31	0.31	0.31
	$er(x_k^{ag})$	24.13	24.13	24.13	15.23	15.2	15.2
UPFAG-LBFGS	Iter(k)	2	2	3	67	168	201
	$T(s)$	0	0	0	0.3	0.7	0.8
	$\Psi(x_k^{ag})$	0.488	0.488	0.483	0.31	0.31	0.31
	$er(x_k^{ag})$	24.13	24.13	24.13	15.24	15.23	15.22
UPFAG-fullBB	Iter(k)	2	2	2	258	451	743
	$T(s)$	0	0	0	1.1	1.7	2.6
	$\Psi(x_k^{ag})$	0.484	0.484	0.484	0.31	0.31	0.31
	$er(x_k^{ag})$	24.13	24.13	24.13	15.21	15.22	15.21
UPFAG-GradDescent	Iter(k)	15	34	94	>3000	>3000	>3000
	$T(s)$	0.1	0.1	0.3	–	–	–
	$\Psi(x_k^{ag})$	0.512	0.484	0.482	–	–	–
	$er(x_k^{ag})$	24.92	24.16	24.13	–	–	–
UFAPL	Iter(k)	1	1	6	46	70	101
	$T(s)$	0.1	0.1	0.1	0.4	0.5	0.6
	$\Psi(x_k^{ag})$	0.591	0.591	0.485	0.31	0.31	0.31
	$er(x_k^{ag})$	24.13	24.13	24.13	15.24	15.22	15.22
UAPL	Iter(k)	1	1	6	55	94	116
	$T(s)$	0.1	0.1	0.1	0.7	1.1	1.3

Table 2 continued

$\|g_X(\bar{x}^*)\|^2$		$< 10^{-2}$	$< 10^{-4}$	$< 10^{-6}$	$< 10^{-8}$	$< 10^{-10}$	$< 10^{-12}$
	$\Psi(x_k^{ag})$	0.591	0.591	0.485	0.31	0.31	0.31
	$er(x_k^{ag})$	24.13	24.13	24.13	15.25	15.22	15.22
w1a							
LBFGS	Iter(k)	20	21	748	2714	2738	2747
	$T(s)$	0.2	0.2	0.7	1.9	1.9	1.9
	$\Psi(x_k^{ag})$	0.213	0.209	0.146	0.145	0.144	0.144
	$er(x_k^{ag})$	7.31	7.26	6.44	6.47	6.45	6.45
UPFAG-LBFGS	Iter(k)	12	14	24	34	93	100
	$T(s)$	0	0	0.1	0.1	0.1	0.2
	$\Psi(x_k^{ag})$	0.231	0.186	0.146	0.145	0.144	0.144
	$er(x_k^{ag})$	8.99	7.53	6.42	6.47	6.45	6.46
UPFAG-fullBB	Iter(k)	12	15	24	63	73	95
	$T(s)$	0	0	0	0.1	0.1	0.1
	$\Psi(x_k^{ag})$	0.28	0.175	0.146	0.144	0.144	0.144
	$er(x_k^{ag})$	11.05	7.05	6.45	6.46	6.46	6.45
UPFAG-GradDescent	Iter(k)	20	70	944	>3000	>3000	>3000
	$T(s)$	0	0.1	0.9	–	–	–
	$\Psi(x_k^{ag})$	0.332	0.189	0.145	–	–	–
	$er(x_k^{ag})$	13.27	7.58	6.45	–	–	–
UFAPL	Iter(k)	1	4	17	41	45	54
	$T(s)$	0.1	0.1	0.2	0.3	0.3	0.3
	$\Psi(x_k^{ag})$	0.642	0.157	0.146	0.144	0.144	0.144
	$er(x_k^{ag})$	6.85	6.69	6.45	6.45	6.45	6.45
UAPL	Iter(k)	1	3	14	1915	1941	1949
	$T(s)$	0.1	0.1	0.2	20.2	20.6	20.7
	$\Psi(x_k^{ag})$	0.642	0.201	0.146	0.145	0.144	0.144
	$er(x_k^{ag})$	6.85	6.74	6.48	6.46	6.45	6.45

$$\min_{\|x\|\le a}\left\{\Psi(x) := \frac{1}{m}\sum_{i=1}^{m} l(u_i, v_i; x) + \frac{\lambda}{2}\|x\|_2^2\right\} \tag{4.1}$$

for some $\lambda > 0$, where $l(x; u_i, v_i)$ is either the nonconvex sigmoid loss function [20] i.e.,

$$l(x; u_i, v_i) = 1 - \tanh\left(v_i \langle x, u_i\rangle\right),$$

or the convex squared hinge one i.e.,

$$l(x; u_i, v_i) = \max\left\{0, 1 - v_i \langle x, u_i\rangle\right\}^2.$$

Note that the above loss functions are smooth and have Lipschitz continuous gradients. Hence, (4.1) fits into the setting of problem (1.1) with $\mathcal{X} \equiv 0$.

Here, we use both real and synthetic data sets. For the former, we use some data sets from LIBSVM at http://www.csie.ntu.edu.tw/~cjlin/libsvm. For the latter, we assume that

Table 3 Average required number of iterations (Iter(k)), runtime ($T(s)$), objective value, and classification error found till reaching a desired accuracy for $\|g_X(\bar{x}^*)\|^2$ over 10 random generated instances of the SVM problem with the squared hinge loss function

$\|g_X(\bar{x}^*)\|^2$		$< 10^{-2}$	$< 10^{-4}$	$< 10^{-6}$	$< 10^{-8}$	$< 10^{-10}$	$< 10^{-12}$
$m = 1000, n = 2000$							
LBFGS	Iter(k)	2.8	12.3	20.4	29.7	36.3	44.7
	$T(s)$	0	0.2	0.3	0.3	0.3	0.3
	$\Psi(x_k^{ag})$	0.451	0.0942	0.0891	0.089	0.089	0.089
	$er(x_k^{ag})$	40.84	32.47	32.49	32.55	32.55	32.56
UPFAG-LBFGS	Iter(k)	2.5	11.7	19.2	28.6	34.9	43
	$T(s)$	0	0	0	0.1	0.1	0.1
	$\Psi(x_k^{ag})$	0.477	0.094	0.0891	0.089	0.089	0.089
	$er(x_k^{ag})$	41.57	32.47	32.49	32.55	32.55	32.56
UPFAG-fullBB	Iter(k)	2.9	14.2	31.7	58.5	101	142
	$T(s)$	0	0	0	0.1	0.1	0.2
	$\Psi(x_k^{ag})$	0.539	0.117	0.0891	0.089	0.089	0.089
	$er(x_k^{ag})$	42.37	34.47	32.54	32.55	32.55	32.55
UPFAG-GradDescent	Iter(k)	5.3	25.1	56.2	183.9	372.8	617.7
	$T(s)$	0	0	0.1	0.2	0.4	0.6
	$\Psi(x_k^{ag})$	0.676	0.124	0.0893	0.089	0.089	0.089
	$er(x_k^{ag})$	43.79	34.82	32.53	32.56	32.55	32.56
UFAPL	Iter(k)	2.8	11.3	18.9	28.3	34.9	42.3
	$T(s)$	0	0.1	0.1	0.1	0.2	0.2
	$\Psi(x_k^{ag})$	0.457	0.0948	0.0891	0.089	0.089	0.089
	$er(x_k^{ag})$	41.33	32.43	32.51	32.56	32.55	32.56
UAPL	Iter(k)	2.6	16.9	26.6	34.3	41.2	52.2
	$T(s)$	0.2	1.4	2.4	3.1	3.8	5.1
	$\Psi(x_k^{ag})$	1.14	0.0955	0.0891	0.089	0.089	0.089
	$er(x_k^{ag})$	36.37	32.28	32.5	32.55	32.56	32.56
AG	Iter(k)	5.3	25.1	70	143.7	267	527.8
	$T(s)$	0	0	0.1	0.1	0.2	0.4
	$\Psi(x_k^{ag})$	0.089	0.089	0.089	0.089	0.089	0.089
	$er(x_k^{ag})$	43.79	34.82	32.56	32.55	32.56	32.55
FAPL	Iter(k)	14.3	45.5	96.3	150.2	>3000	>3000
	$T(s)$	0	0.1	0.2	0.3	–	–
	$\Psi(x_k^{ag})$	0.633	0.102	0.0891	0.089	–	–
	$er(x_k^{ag})$	32.55	32.55	32.55	32.55	–	–

each data point (u_i, v_i) is drawn from the uniform distribution on $[0, 1]^n \times \{-1, 1\}$, where $u_i \in \mathbb{R}^n$ is the feature vector and $v_i \in \{-1, 1\}$ denotes the corresponding label. Moreover, we assume that u_i is sparse with 5% nonzero components and $v_i = \text{sign}(\langle \bar{x}, u_i \rangle)$ for some $\bar{x} \in \mathbb{R}^n$ with $\|\bar{x}\| \leq a$. We also consider two different problem sizes as $n = 2000, 8000$ with $m = 1000, 4000$, respectively.

Table 4 Average required number of iterations (Iter(k)), runtime ($T(s)$), objective value, and classification error found till reaching a desired accuracy for $\|g_X(\bar{x}^*)\|^2$ over 10 random generated instances of the SVM problem with the squared hinge loss function

$\|g_X(\bar{x}^*)\|^2$		$< 10^{-2}$	$< 10^{-4}$	$< 10^{-6}$	$< 10^{-8}$	$< 10^{-10}$	$< 10^{-12}$
$m = 4000, n = 8000$							
LBFGS	Iter(k)	4.3	15.1	43.5	61.4	77.8	94.4
	$T(s)$	0.1	0.2	0.2	0.3	0.4	0.5
	$\Psi(x_k^{ag})$	0.467	0.0798	0.0233	0.0232	0.0232	0.0232
	$er(x_k^{ag})$	41.11	35.7	32.27	32.32	32.36	32.36
UPFAG-LBFGS	Iter(k)	4.2	13.8	43.9	62.8	80.2	96.1
	$T(s)$	0	0.1	0.3	0.4	0.5	0.6
	$\Psi(x_k^{ag})$	0.44	0.111	0.0234	0.0232	0.0232	0.0232
	$er(x_k^{ag})$	42.29	37.16	32.21	32.35	32.36	32.36
UPFAG-fullBB	Iter(k)	6	16.8	92	226.7	407.9	585.8
	$T(s)$	0	0.1	0.6	1.1	1.8	2.5
	$\Psi(x_k^{ag})$	0.579	0.187	0.0235	0.0232	0.0232	0.0232
	$er(x_k^{ag})$	43.38	40.29	32.37	32.36	32.36	32.36
UPFAG-GradDescent	Iter(k)	14	44.2	212.9	693.8	1996.4	>3000
	$T(s)$	0.1	0.2	0.8	2.6	7.5	–
	$\Psi(x_k^{ag})$	0.688	0.196	0.024	0.0232	0.0232	–
	$er(x_k^{ag})$	43.92	39.52	32.57	32.36	32.37	–
UFAPL	Iter(k)	4.1	14.5	44.6	63.7	74.4	91
	$T(s)$	0.1	0.1	0.4	0.6	0.7	0.9
	$\Psi(x_k^{ag})$	0.396	0.0648	0.0234	0.0232	0.0232	0.0232
	$er(x_k^{ag})$	38.27	34.07	32.22	32.35	32.36	32.36
UAPL	Iter(k)	4.4	14.4	49	65.5	77	95.9
	$T(s)$	10.6	20.1	53.7	68.4	78.6	95
	$\Psi(x_k^{ag})$	0.354	0.0949	0.0234	0.0232	0.0232	0.0232
	$er(x_k^{ag})$	35.48	33.25	32.12	32.36	32.35	32.36
AG	Iter(k)	14	44.2	261.4	530.6	1054.2	>3000
	$T(s)$	0	0.1	0.7	1.4	2.8	–
	$\Psi(x_k^{ag})$	0.0232	0.0232	0.0232	0.0232	0.0232	–
	$er(x_k^{ag})$	43.92	39.52	32.51	32.37	32.36	–
FAPL	Iter(k)	16.2	29.9	117	240.7	>3000	>3000
	$T(s)$	0.1	0.2	0.6	1.3	–	–
	$\Psi(x_k^{ag})$	0.579	0.0958	0.0233	0.0232	–	–
	$er(x_k^{ag})$	32.37	32.37	32.37	32.37	–	–

In our experiments, we set $\lambda = 1/m$, $a = 50$. The initial point is randomly chosen within the ball centered at origin with radius a. For synthetic data sets, we report the average results of running different algorithms over 10 instances for each problem size using large sample sizes of $K = 10000$. Moreover, in order to further assess the quality of the generated

Table 5 Required number of iterations (Iter(k)), runtime ($T(s)$), objective value, and classification error found till reaching a desired accuracy for $\|g_X(\bar{x}^*)\|^2$ over data set a6a of the SVM problem with the squared hinge loss function

$\|g_X(\bar{x}^*)\|^2$		$< 10^{-2}$	$< 10^{-4}$	$< 10^{-6}$	$< 10^{-8}$	$< 10^{-10}$	$< 10^{-12}$
LBFGS	Iter(k)	11	25	51	245	341	475
	$T(s)$	0.2	0.3	0.3	0.5	0.6	0.8
	$\Psi(x_k^{ag})$	0.472	0.427	0.424	0.423	0.423	0.423
	$er(x_k^{ag})$	15.96	15.31	15.22	15.23	15.23	15.23
UPFAG-LBFGS	Iter(k)	11	25	51	245	341	475
	$T(s)$	0.1	0.1	0.2	0.7	0.9	1.2
	$\Psi(x_k^{ag})$	0.472	0.427	0.424	0.423	0.423	0.423
	$er(x_k^{ag})$	15.96	15.31	15.22	15.23	15.23	15.23
UPFAG-fullBB	Iter(k)	23	64	191	606	1828	>3000
	$T(s)$	0.1	0.2	0.6	2	5	–
	$\Psi(x_k^{ag})$	0.473	0.433	0.424	0.423	0.423	–
	$er(x_k^{ag})$	15.96	15.29	15.29	15.23	15.23	–
UPFAG-GradDescent	Iter(k)	31	126	395	1493	>3000	>3000
	$T(s)$	0.1	0.3	0.9	3.3	–	–
	$\Psi(x_k^{ag})$	0.516	0.43	0.424	0.423	–	–
	$er(x_k^{ag})$	17.09	15.23	15.22	15.21	–	–
UFAPL	Iter(k)	9	22	86	225	358	494
	$T(s)$	0.2	0.2	0.5	1.2	1.8	2.4
	$\Psi(x_k^{ag})$	0.478	0.427	0.424	0.423	0.423	0.423
	$er(x_k^{ag})$	16.07	15.33	15.22	15.22	15.23	15.23
UAPL	Iter(k)	24	33	59	144	204	311
	$T(s)$	0.3	0.4	0.7	1.7	2.4	3.7
	$\Psi(x_k^{ag})$	0.506	0.426	0.423	0.423	0.423	0.423
	$er(x_k^{ag})$	16.12	15.32	15.21	15.24	15.23	15.23
AG	Iter(k)	31	126	395	1546	>3000	>3000
	$T(s)$	0.1	0.2	0.6	2.5	–	–
	$\Psi(x_k^{ag})$	0.516	0.43	0.424	0.423	–	–
	$er(x_k^{ag})$	17.09	15.23	15.22	15.22	–	–
FAPL	Iter(k)	44	87	327	1477	>3000	>3000
	$T(s)$	0.2	0.3	1	4.7	–	–
	$\Psi(x_k^{ag})$	0.497	0.426	0.423	0.423	–	–
	$er(x_k^{ag})$	15.22	15.22	15.22	15.22	–	–

solutions, we also report the classification error evaluated at the classifier x given by

$$er(x) := \frac{|\{i : v_i \neq \text{sign}(\langle x, u_i \rangle), i = 1, \ldots, K\}|}{K}. \tag{4.2}$$

Real data sets are divided into two categories. Training sets are used to form the loss functions and the results are evaluated using the test sets. Tables 1, 2, 3, 4, 5 and 6 summarize

Table 6 Required number of iterations (Iter(k)), runtime ($T(s)$), objective value, and classification error found till reaching a desired accuracy for $\|g_X(\bar{x}^*)\|^2$ over data set w1a of the SVM problem with the squared hinge loss function

$\|g_X(\bar{x}^*)\|^2$		$< 10^{-2}$	$< 10^{-4}$	$< 10^{-6}$	$< 10^{-8}$	$< 10^{-10}$	$< 10^{-12}$
LBFGS	Iter(k)	6	10	24	36	54	63
	$T(s)$	0.2	0.2	0.2	0.3	0.3	0.3
	$\Psi(x_k^{ag})$	0.322	0.151	0.131	0.131	0.131	0.131
	$er(x_k^{ag})$	9.72	7.05	6.97	6.97	6.98	6.98
UPFAG-LBFGS	Iter(k)	6	11	24	36	44	55
	$T(s)$	0	0	0.1	0.1	0.1	0.1
	$\Psi(x_k^{ag})$	0.298	0.146	0.131	0.131	0.131	0.131
	$er(x_k^{ag})$	9.98	7.05	6.99	6.98	6.98	6.99
UPFAG-fullBB	Iter(k)	6	17	42	108	145	251
	$T(s)$	0	0	0.1	0.1	0.1	0.2
	$\Psi(x_k^{ag})$	0.298	0.149	0.132	0.131	0.131	0.131
	$er(x_k^{ag})$	9.93	7.06	6.97	7.04	6.97	7.01
UPFAG-GradDescent	Iter(k)	19	84	390	1554	>3000	>3000
	$T(s)$	0	0.1	0.3	1.2	–	–
	$\Psi(x_k^{ag})$	0.36	0.158	0.131	0.131	–	–
	$er(x_k^{ag})$	10.38	7.15	6.99	6.97	–	–
UFAPL	Iter(k)	4	9	23	34	50	67
	$T(s)$	0.1	0.1	0.2	0.2	0.3	0.4
	$\Psi(x_k^{ag})$	0.354	0.15	0.131	0.131	0.131	0.131
	$er(x_k^{ag})$	10.33	7.07	6.99	6.97	7.02	7.02
UAPL	Iter(k)	4	12	26	37	46	59
	$T(s)$	0.1	0.2	0.4	0.5	0.6	0.8
	$\Psi(x_k^{ag})$	0.598	0.163	0.131	0.131	0.131	0.131
	$er(x_k^{ag})$	8.23	7.13	6.91	6.96	6.97	7.04
AG	Iter(k)	19	84	390	1431	>3000	>3000
	$T(s)$	0	0.1	0.2	0.9	–	–
	$\Psi(x_k^{ag})$	0.36	0.158	0.131	0.131	–	–
	$er(x_k^{ag})$	10.38	7.15	6.99	7.01	–	–
FAPL	Iter(k)	6	20	75	240	>3000	>3000
	$T(s)$	0	0.1	0.2	0.3	–	–
	$\Psi(x_k^{ag})$	0.253	0.152	0.131	0.131	–	–
	$er(x_k^{ag})$	6.96	6.96	6.96	6.96	–	–

the results of implementing the following algorithms. The LBFGS method only performs the local quasi-Newton search step (2.13), where G_k is the LBFGS [26,27] matrix, without incorporating the acceleration techniques for convex optimization. The UPFAG-LBFGS, UPFAG-fullBB, and UPFAG-GradDescent methods are different variants of Algorithm 1. In particular, the UPFAG-LBFGS applys the LBFGS local search (2.13) and uses the Barzilai-Borwein initial stepsizes (2.12). The UPFAG-fullBB and the UPFAG-GradDescent methods both apply the gradient projection step (2.10), while the former uses the Barzilai-Borwein initial stepsizes (2.12), and the latter tries the stepsize policy in (2.15). The UAPL and the UFAPL methods are also equipped with the LBFGS quai-Newton step (2.13). In all the above algorithms, step (2.13), if used, is only solved inexactly according to (2.14) with $\sigma = 10^{-10}$. In addition to the above algorithms, when the convex loss function is used, we also implement two well-known optimal methods for convex optimization, namely, a variant of the accelerated gradient (AG) method in [11] and the FAPL method in [7]. To make a fair comparison, the generalized projected gradient (2.16) with $\beta_k = 1$ are always used in the stopping condition for all the comparison algorithms.

The following observations can be made from the numerical results. First, for the nonconvex problems, the UPFAG-GradDescent method performs the worst among all the compared algorithms. Second, the UPFAG-LBFGS method outperforms the other two variants of Algorithm 1 showing the effectiveness of the quasi-Newton step (2.13) in comparison to the gradient descent step in (2.10). The LBFGS method is also usually worse than the UPFAG-LBFGS method. Third, the UPFAG-LBFGS and the UFAPL methods have the best and most stable performances over all instances. While both methods have comparable running times, the UFAPL method sometimes outperforms the UPFAG-LBFGS method in terms of number iterations. The UAPL method is also usually comparable to the UFAPL method in terms of number of iterations. However, its running time is much worse as expected due to the existence of two subproblems at each iteration. It is worth noting that the UFAPL method simplifies the subproblems of the UAPL method and hence reduce its running time only when the feasible region is ball-constrained or unconstrained.

In addition, similar observations can be also made on the above-mentioned algorithms when convex loss function is used. Furthermore, it can be observed that by incorporating local quasi-Newton steps, the unified algorithms presented here perform significantly better than the two optimal methods even for solving convex problems, which shows the practical importance of using the local search steps (2.10) and (2.13) especially in a large data setting. Finally, the bundle-level type methods often outperform the accelerated gradient type methods in terms of the iteration number for solving both convex and nonconvex models, indicating the potential of these methods for the situation when the computation of gradients are more expensive than the solution of projection subproblems. This observation has also been particularly made for convex problems without using the local search step (see e.g., [7,15]).

5 Concluding Remarks

In this paper, we extend the framework of uniformly optimal algorithms, currently designed for convex programming, to nonconvex nonlinear programming. In particular, by incorporating a gradient descent step into the framework of uniformly optimal convex programming methods, namely, accelerated gradient and bundle-level type methods, and enforcing the function values evaluated at each iteration of these methods non-increasing, we present unified algorithms for minimizing composite objective functions given by summation of a function f with Hölder continuous gradient and simple convex term over a convex set. We show that these algorithms exhibit the best known convergence rate when f is nonconvex and possess the optimal iteration convergence rate if f turns out to be convex. Therefore, these algorithms allow us to have a unified treatment for nonlinear programming problems regardless of their smoothness level and convexity property. Furthermore, we show that by incorporating quasi-

Newton local search steps, the practical performance of these algorithms can be often greatly improved. Preliminary numerical experiments are also presented to show the performance of our developed algorithms for solving a couple of important nonlinear programming problems arising from machine learning.

References

1. Asmussen, S., Glynn, P.W.: Stochastic Simulation: Algorithm and Analysis. Springer, New York (2000)
2. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**, 183–202 (2009)
3. Becker, Stephen, Fadili, Jalal M.: A quasi-newton proximal splitting method. Adv. Neural Inf. Process. Syst. **25**, 2618–2626 (2012)
4. Ben-Tal, A., Nemirovski, A.S.: Non-Euclidean restricted memory level method for large-scale convex optimization. Math. Program. **102**, 407–456 (2005)
5. Byrd, R.H., Nocedal, J., Schnabel, R.B.: Representations of quasi-newton matrices and their use in limited memory methods. Math. Program. **63**(4), 129–156 (1994)
6. Cartis, C., Gould, N.I.M., Toint, PhL: On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization. SIAM J. Optim. **20**(6), 2833–2852 (2010)
7. Chen, Y., Lan, G., Ouyang, Y., Zhang, W.: Fast bundle-level type methods for unconstrained and ball-constrained convex optimization. Manuscript, University of Florida, Gainesville, FL 32611, USA, December 2014. http://www.optimization-online.org/
8. Fan, J., Li, R.: Variable selection via nonconcave penalized likelihood and its oracle properties. J. Am. Stat. Assoc. **96**, 13481360 (2001)
9. Fu, M.: Optimization for simulation: theory vs. practice. INFORMS J. Comput. **14**, 192–215 (2002)
10. Ghadimi, S., Lan, G.: Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, II: shrinking procedures and optimal algorithms. SIAM J. Optim. **23**, 2061–2089 (2013)
11. Ghadimi, S., Lan, G.: Accelerated gradient methods for nonconvex nonlinear and stochastic optimization. Math. Program. (2015). https://doi.org/10.1007/s10107-015-0871-8
12. Ghadimi, S., Lan, G., Zhang, H.: Mini-batch stochastic approximation methods for constrained nonconvex stochastic programming. Math. Program. (2014). https://doi.org/10.1007/s10107-014-0846-1
13. Hager, W.W., Zhang, H.: Projection on a polyhedron that exploits sparsity. Manuscript, University of Florida and Louisiana State University, Gainesville, FL 32611, USA and Baton Rouge, LA (June 2015)
14. Lan, G.: An optimal method for stochastic composite optimization. Math. Program. **133**(1), 365–397 (2012)
15. Lan, G.: Bundle-level type methods uniformly optimal for smooth and non-smooth convex optimization. Math. Program. **149**(1), 145 (2015)
16. Lan, G.: The complexity of large-scale convex programming under a linear optimization oracle. Manuscript, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA (June 2013). http://www.optimization-online.org
17. Law, A.M.: Simulation Modeling and Analysis. McGraw Hill, New York (2007)
18. Lemaréchal, C., Nemirovski, A.S., Nesterov, Y.E.: New variants of bundle methods. Math. Program. **69**, 111–148 (1995)
19. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online dictionary learning for sparse coding. In: ICML, pp. 689–696 (2009)
20. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent in function space. In: Proceedings of the NIPS, vol. 12, pp. 512–518 (1999)
21. Nemirovski, A.S., Yudin, D.: Problem Complexity and Method Efficiency in Optimization. Wiley-Interscience Series in Discrete Mathematics. Wiley, XV, New York (1983)
22. Nesterov, Y.E.: A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. Dokl. Acad. Nauk SSSR **269**, 543–547 (1983)
23. Nesterov, Y.E.: Introductory Lectures on Convex Optimization: A Basic Course. Kluwer, Boston (2004)
24. Nesterov, Y.E.: Gradient methods for minimizing composite objective functions. Math. Program. Ser. B **140**, 125–161 (2013)
25. Nesterov, Y.E.: Universal gradient methods for convex optimization problems. Math.Program. Ser. A (2014). https://doi.org/10.1007/s10107-014-0790-0

26. Nocedal, J.: Updating quasi-newton matrices with limited storage. Math. Comput. **35**(151), 773–782 (1980)
27. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
28. Devolder, O., Glineur, F., Nesterov, Y.E.: First-order methods of smooth convex optimization with inexact oracle. CORE, Université catholique de Louvain, Louvain-la-Neuve, Belgium, Manuscript (December 2010)
29. Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization. University of Washington, Seattle, Manuscript (May 2008)
30. Yashtini, M.: On the global convergence rate of the gradient descent method for functions with Hölder continuous gradients. Optim. Lett. (2015). https://doi.org/10.1007/s11590-015-0936-x

Affiliations

Saeed Ghadimi[1] · Guanghui Lan[2] · Hongchao Zhang[3]

✉ Guanghui Lan
george.lan@isye.gatech.edu
http://pwp.gatech.edu/guanghui-lan/

Saeed Ghadimi
sghadimi@princeton.edu

Hongchao Zhang
hozhang@math.lsu.edu
https://www.math.lsu.edu/ hozhang

1 Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ 08544, USA

2 School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

3 Department of Mathematics, Louisiana State University, Baton Rouge, LA 70803, USA