A Composite Randomized Incremental Gradient Method

Junyu Zhang [1] **Lin Xiao** [2]

Abstract

We consider the problem of minimizing the composition of a smooth function (which can be nonconvex) and a smooth vector mapping, where both of them can be express as the average of a large number of components. We propose a composite randomized incremental gradient method based on SAGA type of construction. The gradient sample complexity of our method matches that of several recently developed methods based on SVRG in the general case. However, for structured problems where linear convergence rates can be obtained, our method can be much better for ill-conditioned problems. In addition, when the finite-sum structure only appear for the inner mapping, the sample complexity of our method is the same as that of SAGA for minimizing finite sum of smooth nonconvex functions, despite the additional outer composition and the stochastic composite gradients being biased in our case.

1. Introduction

We consider composite optimization problems of the form

$$\underset{x\in\mathbf{R}^d}{\text{minimize}} \quad f\left(\frac{1}{n}\sum_{i=1}^{n} g_i(x)\right) + r(x), \tag{1}$$

where $f : \mathbf{R}^p \to \mathbf{R}$ is a smooth and possibly nonconvex function, each $g_i : \mathbf{R}^d \to \mathbf{R}^p$ is a smooth vector mapping for $i = 1, \ldots, n$, and $r : \mathbf{R}^d \to \mathbf{R} \cup \{\infty\}$ is a convex but possibly nonsmooth function. Problem (1) is a special case of the following more general problem with composite finite sum (average) structure:

$$\underset{x\in\mathbf{R}^d}{\text{minimize}} \quad \frac{1}{m}\sum_{j=1}^{m} f_j\left(\frac{1}{n}\sum_{i=1}^{n} g_i(x)\right) + r(x), \tag{2}$$

where each $f_j : \mathbf{R}^p \to \mathbf{R}$ is smooth and can be nonconvex. Such problems often arise as finite sample approximations of the stochastic composite optimization problem

$$\underset{x\in\mathbf{R}^d}{\text{minimize}} \quad \mathbf{E}_\nu\left[f_\nu\left(\mathbf{E}_\xi[g_\xi(x)]\right)\right] + r(x), \tag{3}$$

where f_ν and g_ξ are parametrized by the random variables ν and ξ of some unknown probability distributions. Many interesting tasks in reinforcement learning (e.g., Sutton & Barto, 1998) and risk-averse learning can be formulated as optimization problems with such composite structure. Algorithms for solving such problems have received lots of attention in recent years, for both the stochastic version (e.g., Wang et al., 2017a;b; Wang & Liu, 2016) and the finite-sum version (e.g., Huo et al., 2018; Lian et al., 2017; Lin et al., 2018; Liu et al., 2018).

In this paper, we focus on randomized algorithms for solving problem (1). While our algorithms and complexity results all extend to the general case (2), focusing on (1) greatly simplifies the presentation and also make the main ideas and analysis much more clear. Nevertheless, our results for the general case are given in the supplementary materials.

Another reason for us to focus on (1) is that most of the interesting applications that we know can be put into this form. A well-known example is the policy evaluation problem in reinforcement learning (RL). With linear value function approximation, it can be formulated as

$$\underset{x\in\mathbf{R}^d}{\text{minimize}} \quad \|\mathbf{E}[A]x - \mathbf{E}[b]\|^2, \tag{4}$$

where A and b are random matrix and vector generated by the transition probability matrix and rewards of a Markov decision process (MDP) (e.g., Dann et al., 2014). A finite sample version of (4) is in the form of (1) with $f(\cdot) = \|\cdot\|^2$.

A more interesting example is risk-averse optimization, which has many applications in RL and financial mathematics. We consider a general formulation of mean-variance trade-off:

$$\underset{x\in\mathbf{R}^d}{\text{maximize}} \quad \frac{1}{n}\sum_{j=1}^{n} h_j(x) - \frac{\lambda}{n}\sum_{j=1}^{n}\left(h_j(x) - \frac{1}{n}\sum_{i=1}^{n} h_i(x)\right)^2, \tag{5}$$

where each $h_j(x) : \mathbf{R}^d \to \mathbf{R}$ is a reward function. The goal of Problem (5) is to maximize the average reward with a

[1]Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, Minnesota, USA. [2]Microsoft Research, Redmond, Washington, USA. Correspondence to: Junyu Zhang <zhan4393@umn.edu>, Lin Xiao <lin.xiao@microsoft.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

penalty on the variance. If we use the following mappings:

$$g_i(x) : \mathbf{R}^d \to \mathbf{R}^{d+1} = \begin{bmatrix} x \\ h_i(x) \end{bmatrix}, \quad (6)$$

$$f_j(y,z) : \mathbf{R}^{d+1} \to \mathbf{R} = -h_j(y) + \lambda\big(h_j(y) - z\big)^2, \quad (7)$$

then problem (5) can be put into the form of (2) with $m = n$ and $r(x) \equiv 0$. However, note that

$$\frac{1}{n}\sum_{j=1}^{n}\left(h_j(x) - \frac{1}{n}\sum_{i=1}^{n} h_i(x)\right)^2 = \frac{1}{n}\sum_{i=1}^{n} h_i^2(x) - \left(\frac{1}{n}\sum_{i=1}^{n} h_i(x)\right)^2.$$

This allows us to reformulate problem (5) into the form of (1) by using the mappings

$$g_i(x) : \mathbf{R}^d \to \mathbf{R}^2 = \begin{bmatrix} h_i(x) \\ h_i^2(x) \end{bmatrix}, \quad (8)$$

$$f(y,z) : \mathbf{R}^2 \to \mathbf{R} = -y + \lambda y^2 - \lambda z. \quad (9)$$

This leads to a simpler structure and much lower intermediate dimension, i.e., $p = 2$ instead of $p = d + 1$ as in (6) and (7).

Besides these applications, structured problems such as (1) and (2) are of independent interest for research on randomized algorithms. Let's denote the Jacobian matrix of $g_i(x)$ by $g_i'(x)$, which is a matrix in $\mathbf{R}^{p\times d}$. Due to the composition of an expectation inside a nonlinear function, it is very hard to get an unbiased estimation of the gradient

$$\nabla\left(f\left(\frac{1}{n}\sum_{i=1}^{n} g_i(x)\right)\right) = \left(\frac{1}{n}\sum_{i=1}^{n} g_i'(x)\right)^T f\left(\frac{1}{n}\sum_{i=1}^{n} g_i(x)\right).$$

Specifically, let $\mathcal{S} \subseteq \{1,\ldots,n\}$ be a random subset and

$$\tilde{g}(x) = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} g_i(x), \qquad \tilde{g}'(x) = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} g_i'(x).$$

Then $\big(\tilde{g}'(x)\big)^T f(\tilde{g}(x))$ is always a biased estimate of the gradient, unless one is willing to calculate the full average $(1/n)\sum_{i=1}^{n} g_i(x)$. Such difficulties often arise in optimizing more sophisticated objective functions other than the empirical risk (e.g., Chaudhari et al., 2016; Hazan et al., 2016; Gulcehre et al., 2016; Mobahi & Fisher, 2015). As a simplest model, the analysis of randomized algorithms for (1) may provide insights into more general problems.

1.1. Related Work

Wang et al. (2017a) considered the problem of minimizing $F(x) \triangleq \mathbf{E}_v\big[f_v\left(\mathbf{E}_\xi[g_\xi(x)]\right)\big]$, i.e., problem (3) with $r \equiv 0$. They derived algorithms to find ϵ-optimal solutions[1] with sample complexities $O(\epsilon^{-4})$, $O(\epsilon^{-3.5})$ and $O(\epsilon^{-1.25})$ for the smooth nonconvex case, smooth convex case and smooth strongly convex case respectively. Wang et al. (2017b) considered problem (3) with nontrivial $r(x)$ and obtained improved sample complexity of $O(\epsilon^{-2.25})$ for smooth nonconvex case and $O(\epsilon^{-2})$ and $O(\epsilon^{-1})$ for the general convex and strongly convex cases respectively.

For the finite-sum problem (2), several authors applied the variance reduction technique of SVRG (Johnson & Zhang, 2013; Xiao & Zhang, 2014) to obtain improved sample complexities. Lian et al. (2017) considered problem (2) when the objective function is strongly convex and $r(x) = 0$. They derived two algorithms with sample complexities $O((m+n+\kappa^3)\log(\frac{1}{\epsilon}))$ and $O((m+n+\kappa^4)\log(\frac{1}{\epsilon}))$ respectively, where κ is some suitably defined condition number. Huo et al. (2018) developed algorithms based on the SVRG scheme to obtain an $O(m+n+(m+n)^{2/3}\epsilon^{-1})$ sample complexity for the smooth nonconvex case and an $O((m+n+\kappa^3)\log(\frac{1}{\epsilon}))$ sample complexity for strongly convex problems with nonsmooth r. This problem was also studied by Yu & Huang (2017) and they proposed an ADMM style algorithm with complexity $O((m+n+\kappa^4)\log(\frac{1}{\epsilon}))$ in the strongly convex case. More recently, Lin et al. (2018) and Liu et al. (2018) made further improvements over the complexity under various conditions.

1.2. Contributions and Outline

Most previous work on solving the problems (1) and (2) are based on the variance reduction scheme of SVRG. In this paper, we propose a composite randomized increment gradient method using the scheme of SAGA (Defazio et al., 2014). We show that it attains the same sample complexity as the methods based on SVRG in general, but for structured problems that allow linear convergence, we obtain improved complexity bounds than previous methods based on SVRG.

In Section 2, we present the C-SAGA method for solving problem (1). In Section 3, we show that it has $O(n+n^{2/3}\epsilon^{-1})$ sample complexity if the composite part is smooth and r is convex. In Section 4, we show that with additional conditions, i.e., gradient dominant or optimally strongly convex condition, an $O((n+\kappa n^{2/3})\log(\frac{1}{\epsilon}))$ sample complexity can be obtained. We provide numerical experiments in Section 5.

We emphasize that these complexities are the same as those obtained by SVRG-type of methods for solving the problem

$$\underset{x\in\mathbf{R}^d}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} g_i(x) + r(x). \quad (10)$$

where each $g_i : \mathbf{R}^d \to \mathbf{R}$ is a scalar mapping (Allen-Zhu & Hazan, 2016; Reddi et al., 2016a;b; Lei et al., 2017). These results indicate that the additional smooth composition over finite-sum problems does not incur higher complexity.

Our algorithm and results extend to the general case (2), with sample complexity $O(m+n+(m+n)^{2/3}\epsilon^{-1})$ for smooth

[1] Here by ϵ-optimal solution, we mean some $x \in \mathbf{R}^d$ that satisfies $\mathbf{E}[F(x) - F^\star] < \epsilon$ in the convex case, where $F^\star = \inf_x F(x)$, and $\mathbf{E}[\|\nabla F(x)\|^2] < \epsilon$ in the smooth nonconvex case.

nonconvex problems and $O((m+n+\kappa(m+n)^{2/3})\log(\frac{1}{\epsilon}))$ under gradient dominant or strongly convex conditions. Compared with the $O((m+n+\kappa^3)\log(\frac{1}{\epsilon}))$ complexity of Lian et al. (2017) and Huo et al. (2018), our result provides (much) better bound as long as $\kappa > n^{1/3}$. The details for the general case are given in the supplementary materials.

2. The Composite SAGA Method

For the ease of presentation, we define the following functions related to the objective in (1):

$$g(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} g_i(x), \quad F(x) \triangleq f(g(x)), \tag{11}$$

$$\Phi(x) \triangleq F(x) + r(x) = f(g(x)) + r(x). \tag{12}$$

First we review the proximal gradient method for solving problems of the form

$$\underset{x\in\mathbf{R}^d}{\text{minimize}} \quad \big\{\Phi(x) = F(x) + r(x)\big\}, \tag{13}$$

where F is smooth and r is convex and may be nonsmooth. The proximal operator of r with parameter η is defined as

$$\mathbf{prox}_r^{\eta}(x) := \arg\min_y \Big\{r(y) + \frac{1}{2\eta}\|y-x\|^2\Big\}. \tag{14}$$

We assume that the function r is relatively simple, meaning that its proximal operator has a close-form solution or can be computed efficiently. The proximal gradient method for solving problem (13) is

$$x^{t+1} = \mathbf{prox}_r^{\eta}\big(x^t - \eta F'(x^t)\big), \tag{15}$$

where $F'(x^t)$ denotes the gradient of F at x^t and η is an appropriate step size (e.g., Nesterov, 2013). For convenience, we define the proximal gradient mapping of Φ as

$$\mathcal{G}(x) \triangleq \frac{1}{\eta}\Big(x - \mathbf{prox}_r^{\eta}\big(x - \eta F'(x)\big)\Big). \tag{16}$$

As a result, the proximal gradient method (15) can be written as $x^{t+1} = x^t - \eta\,\mathcal{G}(x^t)$. Notice that when $r(x) \equiv 0$, we have $\mathcal{G}(x) \equiv F'(x)$ for any $\eta > 0$.

For problem (1) and any x^* generated by some randomized algorithm, we call x^* an ϵ-stationary point in expectation if

$$\mathbf{E}\big[\|\mathcal{G}(x^*)\|^2\big] \le \epsilon. \tag{17}$$

The aim of an efficient (randomized) algorithm is to find such a solution with low sample complexities of the individual functions g_i and their Jacobian g_i' (the total number of times they need to be evaluated). For the batch proximal gradient method (15), its iteration complexity is $O(L_F/\epsilon)$ (e.g., Nesterov, 2004), where L_F is the Lipschitz constant of F'. This translates into a sample complexity of $O(L_F n/\epsilon)$ of the components g_i and g_i'. Our goal is to develop a randomized algorithm that has lower sample complexity.

Algorithm 1 Composite SAGA (C-SAGA)

input: initial point $x^0 \in \mathbf{R}^d$, initial reference points α_i^0 for $i = 1,\dots,n$, and step size $\eta > 0$

Initialize average mapping and Jacobian:

$$Y_0 = \frac{1}{n}\sum_{i=1}^{n} g_i(\alpha_i^0), \quad Z_0 = \frac{1}{n}\sum_{i=1}^{n} g_i'(\alpha_i^0)$$

for $t = 0, ..., T-1$ **do**

Sample with replacement a subset $\mathcal{S}_t \subset \{1,...,n\}$ uniformly at random, with $|\mathcal{S}_t| = s$.

Compute $g_j(x^t)$ and $g_j'(x^t)$ for all $j \in \mathcal{S}_t$ and let

$$y_t = Y_t + \frac{1}{s}\sum_{j\in\mathcal{S}_t}\big(g_j(x^t) - g_j(\alpha_j^t)\big) \tag{18}$$

$$z_t = Z_t + \frac{1}{s}\sum_{j\in\mathcal{S}_t}\big(g_j'(x^t) - g_j'(\alpha_j^t)\big) \tag{19}$$

Let $\tilde{\nabla}F(x^t) = z_t^T f'(y_t)$ and

$$x^{t+1} = \mathbf{prox}_r^{\eta}\big(x^t - \eta\tilde{\nabla}F(x^t)\big) \tag{20}$$

Update reference points:

$$\alpha_j^{t+1} = \begin{cases} x^t & \text{if } j \in \mathcal{S}_t \\ \alpha_j^t & \text{otherwise} \end{cases}$$

Update average mapping and Jacobian:

$$Y_{t+1} = Y_t + \frac{1}{n}\sum_{j\in\mathcal{S}_t}\big(g_j(x^t) - g_j(\alpha_j^t)\big) \tag{21}$$

$$Z_{t+1} = Z_t + \frac{1}{n}\sum_{j\in\mathcal{S}_t}\big(g_j'(x^t) - g_j'(\alpha_j^t)\big) \tag{22}$$

end for

output: Randomly choose $t_* \in \{1,...,T\}$ and output x^{t_*}

We propose to use a randomly selected subset of the functions g_i during each iteration to approximate the full gradient $F'(x) = (g'(x))^T f'(g(x))$. As we pointed out in the introduction, one can easily get an unbiased estimation of $g'(x)$ by $\frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} g_j'(x)$ with $\mathcal{S}$ being a uniformly sampled subset of $\{1,\dots,n\}$. However, an unbiased estimate of $F'(x) = (g'(x))^T f'(g(x))$ cannot be constructed without knowing $g(x)$. Therefore, we have to construct sufficiently accurate estimates of $g(x)$ and $g'(x)$ at the same time and also deal with the bias in estimating $F'(x)$.

In Algorithm 1, we propose C-SAGA, a composite randomized incremental gradient method that employs the estimation scheme of SAGA (Defazio et al., 2014). At each

iteration t, we maintain estimates of $g(x^t)$ and $g'(x^t)$ by

$$Y_t = \frac{1}{n}\sum_{i=1}^{n} g_i(\alpha_i^t) \quad \text{and} \quad Z_t = \frac{1}{n}\sum_{i=1}^{n} g_i'(\alpha_i^t) \tag{23}$$

respectively, where α_i^t are some reference points associated with g_i at iteration t. However, we do not use them directly to estimate $F'(x^t)$, i.e., instead of using $Z_t^T f(Y_t)$, we use $z_t^T f(y_t)$ to estimate $F'(x^t)$, where z_t and y_t are constructed as in (18) and (19) respectively. We define the notation

$$\tilde{\nabla} F(x^t) \triangleq z_t^T f(y_t), \tag{24}$$

and use it to replace $F'(x^t)$ in the proximal gradient method (15), which results in the update formula (20). Then we use the standard SAGA scheme to update the reference points α_j^{t+1} as well as Y_t and Z_t as in (21) and (22).

If each g_i is a scalar mapping, i.e., $g_i : \mathbf{R}^{d_1} \to \mathbf{R}$, and $f(\cdot) = id(\cdot)$ is the scalar identity mapping, then Problem (1) becomes to the standard finite sum optimization problem in (10). In this case, $\tilde{\nabla} F(x^t) = z_t^T f'(y_t) = z_t^T$, hence the update of y_t and Y_t will no longer be necessary. Then Algorithm 1 becomes the standard (minibatch) SAGA algorithm.

We note that Reddi et al. (2016a;b) have developed extensions of SAGA for solving the finite-sum problem (10) with smooth nonconvex functions g_i. However, their method requires two independent sets of samples of size s during each iteration: one for updating the counter part of $\tilde{\nabla} F(x^t) = z_t^T$, and the other set is used for updating the reference points α_j^t and Z_t. In contrast, our method avoids such "double sampling" scheme, and uses only one set of random samples to update both quantities. It reduces to the original SAGA scheme when applied to solve problems (10) with even nonconvex g_i's. Nevertheless, it attains the same sample complexity as the methods by Reddi et al. (2016a;b).

3. Convergence Analysis: Sublinear Rates

We make the following assumption concerning the functions g_i, f, r and g defined in (11).

Assumption 1. *(smoothness assumption)*

- *Each $g_i : \mathbf{R}^d \to \mathbf{R}^p$, for $i = 1, \ldots, n$, is a C^1-smooth vector mapping. In particular, each g_i is ℓ_g-Lipschitz and its Jacobian matrix g_i' is L_g-Lipschitz. Consequently, g is ℓ_g-Lipschitz and g' is L_g-Lipschitz.*
- *The function $f : \mathbf{R}^p \to \mathbf{R}$ is C^1-smooth with f being ℓ_f-Lipschitz and its gradient f' being L_f-Lipschitz.*
- *The function $r : \mathbf{R}^d \to \mathbf{R} \cup \{\infty\}$ is convex and can be nonsmooth.*

As a result of Assumption 1, the gradient of F defined in (11), denoted as F', is L_F-Lipschitz continuous with

$$L_F = \ell_g^2 L_f + \ell_f L_g.$$

We also define the constant

$$G_0 = 18\ell_g^4 L_f^2 + 2\ell_f^2 L_g^2 = O(L_F^2).$$

Theorem 1. *Suppose Assumption 1 holds and we choose $\alpha_i^0 = x^0$ for all $i = 1, \ldots, n$. Let the sequence $\{x^t\}_{t=0}^T$ be generated by Algorithm 1 and t_* is chosen uniformly at random from $\{1, \ldots, T\}$.*

1. *If the batch size $s = 1$, then we can choose the step size*

$$\eta = \frac{1}{4n\sqrt{3G_0}} = O\left(\frac{1}{nL_F}\right)$$

to obtain

$$\mathbf{E}[\|\mathcal{G}(x^{t_*})\|^2] \le \frac{32n\sqrt{3G_0}}{T}\mathbf{E}\left[\Phi(x^0) - \Phi(x^T)\right].$$

Therefore, finding x^{t_*} with $\mathbf{E}[\|\mathcal{G}(x^{t_*})\|^2] \le \epsilon$ requires $O(L_F n/\epsilon)$ function and Jacobian evaluations.

2. *If we choose the batch size $s = n^{2/3}$, and the step size*

$$\eta = \frac{1}{L_F + \sqrt{L_F^2 + 48G_0}} = O\left(\frac{1}{L_F}\right),$$

then

$$\mathbf{E}[\|\mathcal{G}(x^{t_*})\|^2] \le \frac{8}{T\eta}\mathbf{E}\left[\Phi(x^0) - \Phi(x^T)\right] = O\left(\frac{L_F}{T}\right).$$

As a result, finding x^{t_*} with $\mathbf{E}[\|\mathcal{G}(x^{t_*})\|^2] \le \epsilon$ requires $O(n + L_F n^{2/3}/\epsilon)$ function and Jacobian evaluations.

Note that for the case $s = 1$, the sample complexity is $O(L_F n/\epsilon)$, which matches the sample complexity of full-gradient descent method. This is the first result in stochastic composite optimization that can match the performance of full-gradient descent by taking just one sample per iteration. By using $s = n^{2/3}$ and large step size $\eta = O(1/L_F)$, we obtain the improved sample complexity $O(n + L_F n^{2/3}/\epsilon)$.

3.1. Outline of Analysis

Here we give an outline of the proof for Theorem 1, emphasizing the key ideas and steps. Detailed proofs of individual lemmas are given in the supplementary materials.

First, as we explained before, y_t and z_t are unbiased estimates of $g(x^t)$ and $g'(x^t)$ respectively, but $\tilde{\nabla} F(x^t) = z_t^T f'(y_t)$ is a biased estimate of $F'(x^t) = (g'(x^t))^T f'(x^t)$. The following lemma bounds the variances of the unbiased estimates and also the squared bias for estimating $F'(x^t)$.

Lemma 1. *Suppose $\{x^t\}$ and $\{\alpha_j^t\}$ for $j = 1, \ldots, n$ are generated by Algorithm 1. Then we have*

$$\mathbf{E}[y_t | x^t] = g(x^t),$$

$$\mathbf{E}[\|y_t - g(x^t)\|^2 | x^t] \le \frac{\ell_g^2}{s}\frac{1}{n}\sum_{j=1}^{n}\|x^t - \alpha_j^t\|^2,$$

where $\mathbf{E}[\cdot|x^t]$ *denotes conditional expectation given* x^t, *and*

$$\mathbf{E}[z_t|x^t] = g'(x^t),$$
$$\mathbf{E}[\|z_t - g'(x^t)\|^2|x^t] \le \frac{L_g^2}{s}\frac{1}{n}\sum_{j=1}^n \|x^t - \alpha_j^t\|^2.$$

In addition, the bias in estimating $F'(x^t)$ *can be bounded as*

$$\mathbf{E}\left[\|\tilde{\nabla}F(x^t) - F'(x^t)\|^2|x^t\right] \le \frac{G_0}{s}\frac{1}{n}\sum_{j=1}^n \|x^t - \alpha_j^t\|^2.$$

In order to quantify the optimality of x^t, a proper metric is the norm of the proximal gradient mapping

$$\mathcal{G}(x^t) = \frac{1}{\eta}(x^t - \hat{x}^{t+1}),$$

where

$$\hat{x}^{t+1} = \mathbf{prox}_r^\eta\left(x^t - \eta F'(x^t)\right). \tag{25}$$

Eventually, we expect our algorithm to return a point x with $\mathbf{E}[\|\mathcal{G}(x)\|^2] \le \epsilon$. However, in the algorithm, only the approximate proximal gradient mapping

$$\tilde{\mathcal{G}}(x^t) := \frac{1}{\eta}(x^t - x^{t+1}),$$

where x^{t+1} is given in (20), can be computed. Thus we will have to ensure the closeness between $\mathcal{G}(x^t)$ and $\tilde{\mathcal{G}}(x^t)$. This result is given in the next lemma.

Lemma 2. *Let* $\hat{x}^{t+1}$ *be defined according to* (25), *then*

$$\frac{1}{\eta}\|x^{t+1} - \hat{x}^{t+1}\|^2 \le \eta\|\tilde{\nabla}F(x^t) - F'(x^t)\|^2$$

and

$$\mathbf{E}\left[\|\mathcal{G}(x^t)\|^2\right] \le 2\mathbf{E}\left[\|\tilde{\mathcal{G}}(x^t)\|^2\right] + \frac{2G_0}{s}\mathbf{E}\left[\frac{1}{n}\sum_{j=1}^n \|x^t - \alpha_j^t\|^2\right].$$

Using the two lemmas above, we can show the next result, which quantifies the expected descent over the iterations.

Lemma 3. *If the sequence* $\{x^t\}$ *is generated by Algorithm 1, then the following descent result holds:*

$$\mathbf{E}[\Phi(x^{t+1})] \le \mathbf{E}[\Phi(x^t)] - \left(\frac{\eta}{2} - \frac{L_F\eta^2}{2}\right)\mathbf{E}\left[\|\tilde{\mathcal{G}}(x^t)\|^2\right] + \frac{G_0\eta}{2s}\mathbf{E}\left[\frac{1}{n}\sum_{j=1}^n \|x^t - \alpha_j^t\|^2\right]. \tag{26}$$

To study the convergence of Algorithm 1, we also need to construct the following Lyapunov function:

$$R_t = \mathbf{E}\left[\Phi(x^t) + c_t \cdot \frac{1}{n}\sum_{j=1}^n \|x^t - \alpha_j^t\|^2\right] \tag{27}$$

where the coefficients c_t satisfy $c_T = 0$ and the recursion

$$c_t = c_{t+1}(1-p)(1+\beta) + \frac{3G_0}{4s}\eta\,. \tag{28}$$

The coefficient $p = 1 - (1 - \frac{1}{n})^s \ge \frac{s}{2n}$ is the probability that an index j is chosen to get into the set $\mathcal{S}_t$, and $\beta > 0$ is an arbitrary coefficient that will be determined later. The following lemma bounds the evolution of the distances between x^t and the reference points α_i^t.

Lemma 4. *Suppose* $\{x^t\}$ *and* $\{\alpha_j^t\}$ *for* $i = 1, \ldots, n$ *are generated by Algorithm 1, then the following result holds:*

$$\mathbf{E}\left[\frac{1}{n}\sum_{j=1}^n \|x^{t+1} - \alpha_j^{t+1}\|^2\right] \le \left(1 + \frac{1}{\beta}\right)\mathbf{E}\left[\|x^{t+1} - x^t\|^2\right] + (1-p)(1+\beta)\mathbf{E}\left[\frac{1}{n}\sum_{j=1}^n \|x^t - \alpha_j^t\|^2\right],$$

where $\beta > 0$ *is an arbitrary constant appearing in* (28).

Reddi et al. (2016a;b) derived a sharper bound, replacing the coefficient $(1 + 1/\beta)$ above by $(1 + (1-p)/\beta)$, but required two independent set of samples of size s for updating z_t and α^t respectively. Our bound in Lemma 4 is slightly looser, but it only requires one set of samples and does not deteriorate the final complexity. Combining Lemma 3 and Lemma 4, we have the following result.

Lemma 5. *Let the Lyapunov function* R_t *and the coefficients* c_t *be defined according to* (27) *and* (28) *respectively, then*

$$\mathbf{E}\left[\gamma\|\tilde{\mathcal{G}}(x^t)\|^2 + \frac{G_0\eta}{4s}\frac{1}{n}\sum_{j=1}^n \|x^t - \alpha_j^t\|^2\right] \le R_t - R_{t+1}, \tag{29}$$

where

$$\gamma = \min_{0\le t\le T-1}\left\{\frac{\eta}{2} - \frac{L_F\eta^2}{2} - \left(1 + \frac{1}{\beta}\right)c_{t+1}\eta^2\right\}. \tag{30}$$

One last lemma is needed to guarantee that γ is sufficiently large, namely, $\gamma = O(\eta)$.

Lemma 6. *Let* c_t *be defined according to* (28) *and let* γ *be defined in* (30). *If we choose* $\beta = s/(4n)$, *then for any* $0 \le t \le T$, *we have* $c_t \le 3nG_0\eta/s^2$. *As a result, we can set*

1. *either* $s = 1$ *and* $\eta = \dfrac{1}{4n\sqrt{3G_0}} = O\left(\dfrac{1}{nL_F}\right)$,
2. *or* $s = n^{2/3}$ *and* $\eta = \dfrac{1}{L_F + \sqrt{L_F^2 + 48G_0}} = O\left(\dfrac{1}{L_F}\right)$.

Under both set of parameters, we have $\gamma \ge \frac{\eta}{4}$.

Finally, Theorem 1 can be proved using Lemma 5 and Lemma 6; see details in the supplementary materials.

4. Linear Convergence of C-SAGA

In this section, we shall present performance of Algorithm 1 under the gradient-dominant condition and the strong convexity condition, where the algorithm exhibits fast linear convergence.

4.1. Gradient-Dominant Function

First, we analyze the case where $r(x) \equiv 0$ and $F(x)$ is *ν-gradient dominant*. Formally speaking, we make the following assumption.

Assumption 2. *The nonsmooth part $r(x) \equiv 0$ and the smooth part F is ν-gradient dominant, which means there exist some $\nu > 0$ such that*

$$F(x) - \inf_y F(y) \le \frac{\nu}{2}\|F'(x)\|^2, \qquad \forall\, x \in \mathbf{R}^d. \tag{31}$$

Note that strong convexity is a special case of gradient domination, which in turn is a special case of the Polyak-Łojasiewicz condition (e.g., Karimi et al., 2016). More specifically, a strongly convex function with convexity parameter μ is $\frac{1}{\mu}$-gradient dominant (e.g., Nesterov, 2004). We denote

$$\kappa = \nu L_F$$

as the effective condition number.

Reddi et al. (2016a) proposed a restarted SAGA algorithm for solving the finite-sum problem (10) and achieved the sample complexity of $O((n + \kappa n^{2/3})\log(1/\epsilon))$. However, each of their restarts requires synchronization of all the reference points, and the frequent restarts make the algorithm more like a SVRG type of algorithm. Here we show that our C-SAGA method in Algorithm 1, without any modification, achieves the same sample complexity for the more general composite finite-sum problem (1).

We denote the Algorithm 1 (C-SAGA) as a mapping

$$(x^{t_*}, \alpha_1^{t_*}, ..., \alpha_n^{t_*}) = \text{C-SAGA}(x^0, \alpha_1^0, ..., \alpha_n^0, \tau, \eta),$$

where τ is the number of iterations and η is step size. In addition to x^{t_*}, the outputs include the reference points $\alpha_i^{t_*}$ for $i = 1, \ldots, n$ at time t_*, which is chosen uniformly random from $\{1, \ldots, \tau\}$. For the ease of understanding, we imagine running the algorithm in K stages, each with τ iterations and initialized with the previous output and reference points. Since the outputs are snapshots randomly chosen from the previous τ iterations, we don't really need to finish all the τ iterations for each stage. We can just randomly generate an l_k from $\{1, ..., \tau\}$, and then go to the next stage as soon as we reach the l_k-th iteration in the current stage. This is equivalent to partitioning the iterations of Algorithm 1 into some virtually assigned stages.

More precisely, let $\{x^t\}$ and $\{\alpha_1^k, \ldots, \alpha_n^k\}$ be generated by Algorithm 1, and each l_k, for $k = 1, \ldots, K-1$, is an independent uniform sample from $\{1, ..., \tau\}$. Then we assign subsequences $\{\tilde{x}^k\}$ and $\{\tilde{\alpha}_j^k\}$, for $j = 1, \ldots, n$, as follows:

$$\tilde{x}^{k+1} = x^{T_k + l_k}, \text{ and } \tilde{\alpha}_j^{k+1} = \alpha_j^{T_k + l_k},\ j = 1, ..., n, \tag{32}$$

with $\tilde{x}^0 = x^0$, $\tilde{\alpha}_j^0 = \alpha_j^0$, and

$$T_{k+1} = T_k + l_k \text{ for } k = 0, ..., K-1, \text{ with } T_0 = 0\,. \tag{33}$$

Theorem 2. *Suppose Assumption 1 and Assumption 2 holds. Let the sequence $\{x^t\}$ be generated by Algorithm 1 and $\{\tilde{x}^k\}$ is defined by (32) and (33), where each l_k is uniformly sampled from $\{1, ..., \tau\}$. Then if we set the virtual epoch length parameter $\tau = 16\nu/\eta + 24n/s$, then*

$$\mathbf{E}\left[F(\tilde{x}^k) - F(x^*) + \frac{3G_0\eta}{s^2}\sum_{j=1}^n \|\tilde{x}^k - \tilde{\alpha}_j^k\|^2\right]$$
$$\le \frac{1}{2^k}\mathbf{E}\left[F(\tilde{x}^0) - F(x^*) + \frac{3G_0\eta}{s^2}\sum_{j=1}^n \|\tilde{x}^0 - \tilde{\alpha}_j^0\|^2\right].$$

Consequently, in order to guarantee $\mathbf{E}[F(\tilde{x}^k) - F(x^*)] \le \epsilon$:

1. *If we choose $s = 1$ and $\eta = O(1/(nL_F))$, then $\tau = O(n(\nu L_F + 1))$ and the total sample complexity is*

$$O(n(\kappa + 1)\log(1/\epsilon));$$

2. *If we choose $s = n^{2/3}$ and $\eta = O(1/L_F)$, then $\tau = O(\nu L_F + n^{1/3})$ and the total sample complexity is*

$$O(\tau s \log(1/\epsilon)) = O((n + \kappa n^{2/3})\log(1/\epsilon))\,.$$

4.2. Optimally Strongly Convex Function

In this part, we work with general nonsmooth convex proper function $r(x)$ and a *μ-optimally strongly convex* assumption on $\Phi(x) = F(x) + r(x)$. Formally, we assume $\Phi(x)$ satisfies the following assumption.

Assumption 3. *We assume F is convex, and the overall objective function $\Phi(x)$ is μ-optimally strongly convex, i.e.,*

$$\Phi(x) - \Phi(x^*) \ge \frac{\mu}{2}\|x - x^*\|^2, \qquad \forall x \in \mathbf{R}^d \tag{34}$$

for some constant $\mu > 0$, where $x^* = \arg\min_y \Phi(y)$. In addition, we assume r is convex and possibly nonsmooth.

Theorem 3. *Suppose Assumption 1 and Assumption 3 hold and $\alpha_i^0 = x^0$ for all $i = 1, \ldots, n$. Let the sequence $\{x^t\}$ be generated by Algorithm 1. And suppose that $\{\tilde{x}^k\}$ are assigned according to (32) and (33) with l_k sampled uniformly from $\{1, ..., \tau\}$. If we set the virtual epoch length to be $\tau = 28/3\eta\mu + 96n/s$, then*

$$\mathbf{E}\left[\Phi(\tilde{x}^k) - \Phi(x^*)\right] \le \frac{1}{2^k}\mathbf{E}\left[\Phi(x^0) - \Phi(x^*)\right].$$

Consequently, in order to ensure $\mathbf{E}[F(\tilde{x}^k) - F(x^*)] \le \epsilon$:

1. *If $s = 1$ and $\eta = O(1/(nL_F))$, then $\tau = O(n(\kappa + 1))$ and the total sample complexity is*

$$O(n(\kappa + 1)\log(1/\epsilon));$$

2. *If $s = n^{2/3}$ and $\eta = O(1/L_F)$, then $\tau = O(\kappa + n^{1/3})$ and the total sample complexity is*

$$O(\tau s \log(1/\epsilon)) = O((n + \kappa n^{2/3})\log(1/\epsilon))\,.$$

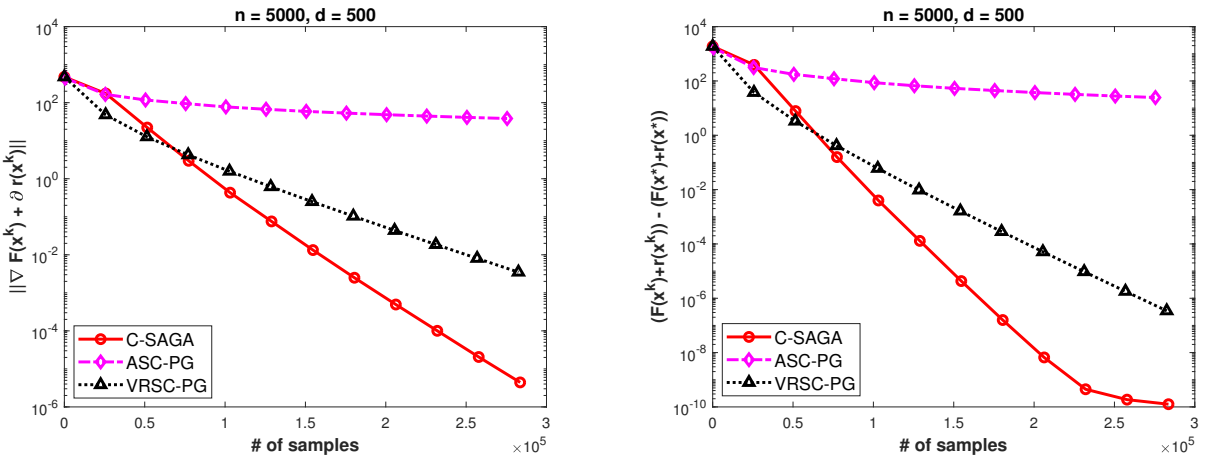

Figure 1. Experiments on the risk-averse portfolio optimization problem

5. Numerical Experiments

In this part, we present numerical experiments for two applications: risk-averse portfolio optimization and policy evaluation for Markov decision processes.

5.1. Risk-Averse Portfolio Optimization

Suppose there are d different assets that we can invest during a time interval of $\{1, ..., n\}$. Let $R_{i,j}$ be the return of asset j at time i, and R_i be the vector consists of $R_{i,1}, \ldots, R_{i,d}$. Then the risk-averse portfolio optimization problem can be formulated as problem (5) by defining

$$h_i(x) = \langle R_i, x\rangle, \qquad i = 1, \ldots, n,$$

where $x \in \mathbf{R}^d$ is the vector of asset allocations. In addition, we also add an ℓ_1 regularization term $r(x) = \beta\|x\|_1$ for some $\beta > 0$, in order to obtain sparse asset allocation.

As we discussed in the introduction, using the mappings defined in (6) and (7), we can transform this problem into the form of (2). We solve this problem via two algorithms that handle the finite-sum structure both inside and outside of the composition: ASC-PG (Wang et al., 2017b) and VRSC-PG (Huo et al., 2018). Alternatively, if we use the mappings defined in (8) and (9), then the problem is transformed into the form of (1). We can solve this formulation with our C-SAGA algorithm.

In our experiments, the reward vectors R_i are first generated as n i.i.d Gaussian random vectors with a random correlation matrix $C = LL^T$, where $L \in \mathbf{R}^{d\times d}$ satisfies $\mathcal{N}(0, 1)$ distribution elementwise. We set the parameter $\lambda = 0.1$ in (5) and the ℓ_1 regularization parameter $\beta = 1$.

We test the algorithms on a randomly generated case with $d = 500, n = 5000$, and plot the experiment results in Figure 1. The curves are averaged over 20 runs and are plotted against the number of samples of the component functions. Both VRSC-PG and C-SAGA use the same step size $\eta = 0.001$ and batch size $s = \lceil n^{2/3}\rceil$. They are chosen from by experimenting with $\{1, 0.1, 0.01, 0.001, 0.0001\}$, and $\eta = 0.001$ works best for VRSC-PG and for C-SAGA.

For ASC-PG, we set its parameters $\alpha_k = 0.001/k$, $\beta_k = 1/k$ (the β_k is different from the sparsity penalty parameter β, see Wang et al. (2017b)). They are hand-tuned to ensure ASC-PG converges fast among a range of other tested parameters.

As shown in Figure 1, ASC-PG is the slowest one due to its lack of variance reduction schemes. Both VRSC-PG and C-SAGA have linear convergence in this case, and C-SAGA is faster than VRSC-PG in terms sample efficiency. This supports our theory that C-SAGA can be more efficient (weaker dependence on the condition number) when linear convergence occurs.

5.2. Policy Evaluation for MDP

Consider a Markov decision process (MDP) with state space $\mathcal{S} = \{1, ..., S\}$. Let the reward associated with transition from state i to state j be $R_{i,j}$. Let $P^\pi \in \mathbf{R}^{S\times S}$ be the transition probability matrix under some fixed policy π. We would like to evaluate the value function $V^\pi : \mathcal{S} \to \mathbf{R}$ under the policy, which satisfies following Bellman equation:

$$V^\pi(i) = \sum_{j=1}^{S} P^\pi_{i,j}(r_{i,j} + \gamma V^\pi(j)) = \mathbf{E}_{j|i}[R_{i,j} + \gamma V^\pi(j)].$$

We apply the linear function approximation $V^\pi(i) \approx \langle \Phi_i, w^*\rangle$ for a given set feature vectors Φ_i (e.g., Dann et al., 2014; Wang et al., 2017b), and would like to compute the optimal vector w^*. This can be formulated as the following problem

$$\underset{w}{\text{minimize}}\ F(w) \triangleq \sum_{i=1}^{S}\left(\langle\Phi_i, w\rangle - \sum_{j=1}^{S} P^\pi_{i,j}(r_{i,j} + \gamma\langle\Phi_j, w\rangle)\right)^2.$$

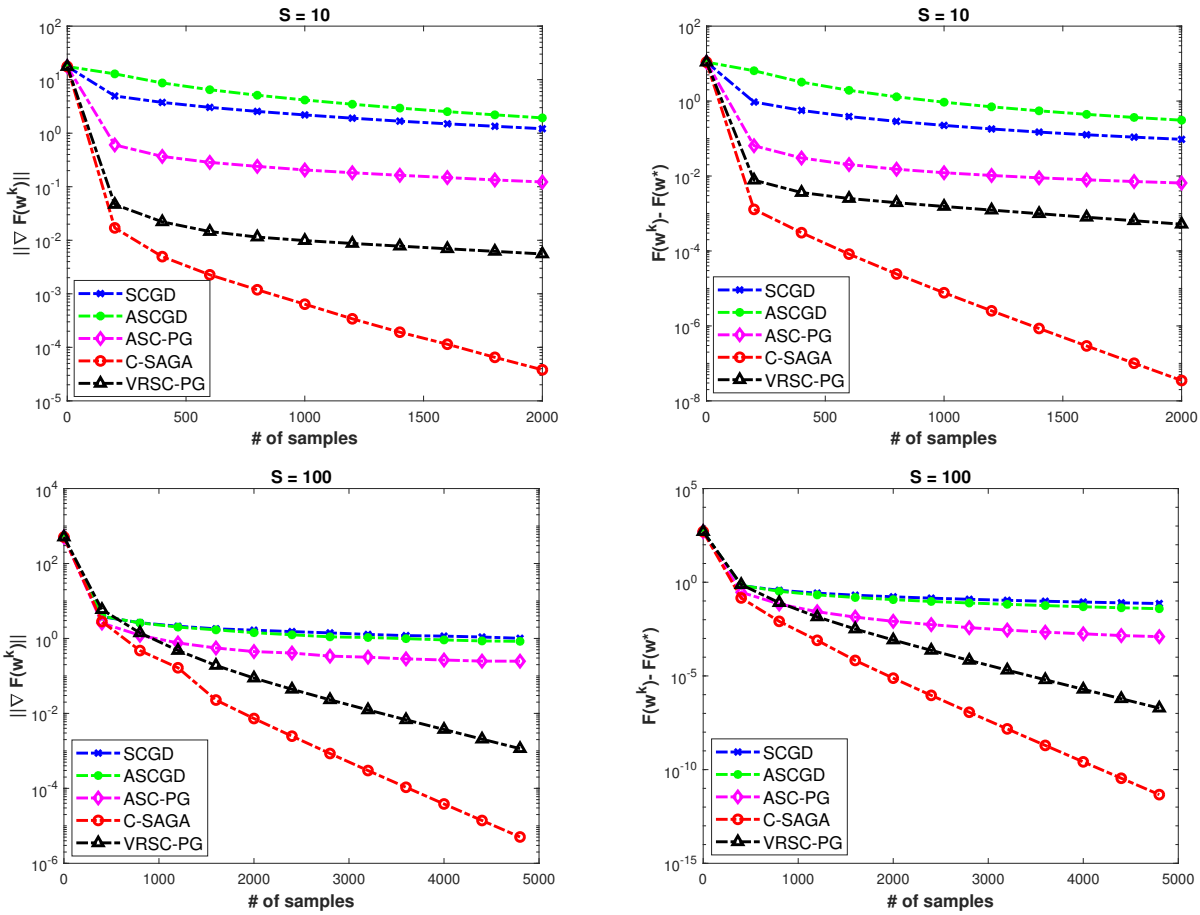

Figure 2. Experiments on MDP policy evaluation problem. Row 1: case with $S = 10$. Row 2: case with $S = 100$.

Let's denote

$$q_i^\pi(w) \triangleq \sum_{j=1}^{S} P_{i,j}^\pi (r_{i,j} + \gamma \langle \Phi_j, w \rangle) = \mathbf{E}_{j|i}[R_{i,j} + \gamma \langle \Phi_j, w \rangle].$$

Then by defining

$$g(w) = \left[\langle \Phi_1, w \rangle, ..., \langle \Phi_S, w \rangle, q_i^\pi(w), ..., q_S^\pi(w) \right]^T$$

and

$$f(y_1, ..., y_S, z_1, ..., z_S) = \|y - z\|^2 = \sum_{i=1}^{S} (y_i - z_i)^2,$$

this problem is transformed into the form of (3). Note that the first S components of g are deterministic, then for SCGD (Wang et al., 2017a), ASCGD (Wang et al., 2017a) and ASC-PG (Wang et al., 2017b), the running average estimation of g is not applied for these components. For the last S components, since they are S independent expectations, the variance reduction technique of both VRSC-PG (Huo et al., 2018) and C-SAGA are applied to each of these components. In the experiments, P^π, Φ and R^π are generated randomly.

Figure 2 shows two experiments with sizes $S = 10$ and $S = 100$ respectively. We plot the objective values and gradient sizes against the samples drawn by the algorithm, and they are shown as averages over 20 runs. For the case where $S = 10$, both VRSC-PG and C-SAGA use the same batch size $s = 1$. C-SAGA takes a step size $\eta = 0.1$, while VRSC-PG takes a stepsize of $\eta = 0.03$, because it diverges under $\eta = 0.1$ and $\eta = 0.03$ seems to work best VRSC-PG.

For $S = 100$, we set $\eta = 0.005$ and batch size $s = 10$ for C-SAGA and VRSC-PG. The step size is chosen as the best among $\{0.1, 0.05, 0.01, 0.005, 0.001\}$. Under $\eta = 0.005$ both VRSC-PG and C-SAGA gain their best performance. For SCGD, we choose $\alpha_k = 0.01k^{-3/4}$ and $\beta_k = 0.1k^{-1/2}$. For ASCGD, $\alpha_k = 0.01k^{-5/7}$ and $\beta_k = 0.1k^{-4/7}$. For ASC-PG, $\alpha_k = 0.01k^{-1/2}$ and $\beta_k = 0.1k^{-1}$. The meaning of these step size parameters can be found in Wang et al. (2017a;b). They are hand-tuned to yield fast convergence. Figure 2 show that C-SAGA has the best performance compared with other methods.

In summary, C-SAGA is very effective for solving composite finite sum problems due to its simple construction and fast convergence. More experiments are presented in the supplementary materials.

References

Allen-Zhu, Z. and Hazan, E. Variance reduction for faster non-convex optimization. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 699–707, 2016.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint, arXiv:1611.01838*, 2016.

Dann, C., Neumann, G., and Peters, J. Policy evaluation with temporal differences: a survey and comparison. *Journal of Machine Learning Research*, 15(1):809–883, 2014.

Defazio, A., Bach, F., and Lacoste-Julien, S. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

Gulcehre, C., Moczulski, M., Visin, F., and Bengio, Y. Mollifying networks. *arXiv preprint, arXiv:1608.04980*, 2016.

Hazan, E., Levy, K. Y., and Shalev-Shwartz, S. On graduated optimization for stochastic non-convex problems. In *International conference on machine learning*, pp. 1833–1841, 2016.

Huo, Z., Gu, B., Jiu, J., and Huang, H. Accelerated method for stochastic composition optimization with nonsmooth regularization. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 3287–3294, 2018.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pp. 315–323, 2013.

Karimi, H., Nutini, J., and Schmidt, M. Linear convergence of gradient method and proximal-gradient methods under the Polyak-łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Database - European Conference, Proceedings*, pp. 795–811, 2016.

Lei, L., Ju, C., Chen, J., and Jordan, M. I. Non-convex finite-sum optimization via scsg methods. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2348–2358. Curran Associates, Inc., 2017.

Lian, X., Wang, M., and Liu, J. Finite-sum composition optimization via variance reduced gradient descent. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1159–1167, 2017.

Lin, T., Fan, C., Wang, M., and Jordan, M. I. Improved oracle complexity for stochastic compositional variance reduced gradient. *arXiv preprint arXiv:1806.00458*, 2018.

Liu, L., Liu, J., Hsieh, C.-J., and Tao, D. Stochastically controlled stochastic gradient for the convex and non-convex composition problem. arXiv preprint, arXiv:1809.02505, 2018.

Mobahi, H. and Fisher, J. W. On the link between gaussian homotopy continuation and convex envelopes. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 43–56. Springer, 2015.

Nesterov, Y. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Boston, 2004.

Nesterov, Y. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

Reddi, S. J., Sra, S., Póczos, B., and Smola, A. Fast incremental method for smooth nonconvex optimization. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 1971–1977. IEEE, 2016a.

Reddi, S. J., Sra, S., Póczos, B., and Smola, A. J. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *Advances in Neural Information Processing Systems*, pp. 1145–1153, 2016b.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Wang, M. and Liu, J. A stochastic compositional gradient method using Markov samples. In *2016 Winter Simulation Conference (WSC)*, pp. 702–713, Dec 2016. doi: 10.1109/WSC.2016.7822134.

Wang, M., Fang, E. X., and Liu, H. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Mathematical Programming*, 161(1-2):419–449, 2017a.

Wang, M., Liu, J., and Fang, E. Accelerating stochastic composition optimization. *Journal of Machine Learning Research*, 18(105):1–23, 2017b.

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Yu, Y. and Huang, L. Fast stochastic variance reduced ADMM for stochastic composition optimization. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3364–3370, 2017.