Asynchronous Proximal Stochastic Gradient Algorithm for Composition Optimization Problems

Pengfei Wang,[1,2] **Risheng Liu,**[3] **Nenggan Zheng,**[1*] **Zhefeng Gong**[4]
[1]Qiushi Academy for Advanced Studies, Zhejiang University, Hangzhou, Zhejiang, China
[2]College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang, China
[3]International School of Information Science & Engineering, Dalian University of Technology, Liaoning, China
[4]Department of Neurobiology, Zhejiang University School of Medicine, Hangzhou, Zhejiang, China
{pfei, zng, zfgong}@zju.edu.cn, rsliu@dlut.edu.cn

Abstract

In machine learning research, many emerging applications can be (re)formulated as the composition optimization problem with nonsmooth regularization penalty. To solve this problem, traditional stochastic gradient descent (SGD) algorithm and its variants either have low convergence rate or are computationally expensive. Recently, several stochastic composition gradient algorithms have been proposed, however, these methods are still inefficient and not scalable to large-scale composition optimization problem instances. To address these challenges, we propose an asynchronous parallel algorithm, named Async-ProxSCVR, which effectively combines asynchronous parallel implementation and variance reduction method. We prove that the algorithm admits the fastest convergence rate for both strongly convex and general nonconvex cases. Furthermore, we analyze the query complexity of the proposed algorithm and prove that linear speedup is accessible when we increase the number of processors. Finally, we evaluate our algorithm Async-ProxSCVR on two representative composition optimization problems including value function evaluation in reinforcement learning and sparse mean-variance optimization problem. Experimental results show that the algorithm achieves significant speedups and is much faster than existing compared methods.

Introduction

Composition optimization problem proposed recently by Wang et al. (Wang, Fang, and Liu 2014) arises in many important applications including reinforcement learning (Wang, Liu, and Tang 2017), statistical learning (Hinton and Roweis 2003), risk management (Shapiro, Dentcheva, and Ruszczyński 2009), and multi-stage stochastic programming (Shapiro, Dentcheva, and Ruszczyński 2009). In this paper, we study the finite-sum scenario for the regularized composition optimization problem whose objective function is the composition of two finite-sum functions plus a possibly nonsmooth regularization term, i.e.,

$$\min_{x\in\mathbb{R}^N} H(x) = f(x) + h(x), \tag{1}$$

where

$$f(x) = \frac{1}{n_1}\sum_{i=1}^{n_1} F_i(\frac{1}{n_2}\sum_{j=1}^{n_2} G_j(x)), \tag{2}$$

where $G_j : \mathbb{R}^N \to \mathbb{R}^M$, $F_i : \mathbb{R}^M \to \mathbb{R}$, and $f : \mathbb{R}^N \to \mathbb{R}$ are continuously differentiable functions. We denote $G(x) := \frac{1}{n_2}\sum_{j=1}^{n_2} G_j(x)$, $y := G(x)$, and $F(y) := \frac{1}{n_1}\sum_{i=1}^{n_1} F_i(y)$. And we call $G(x)$ the inner function and $F(y)$ the outer function. Often $f(x)$ is used as the empirical risk approximation to the composition of two expected-value function $\mathbb{E}_i F_i(\mathbb{E}_j G_j(x))$. The regularization $h : \mathbb{R}^N \to \mathbb{R}\cup\{+\infty\}$ is an extended real-valued closed convex but possibly nonsmooth function, which is often used to constrain the capacity of the hypothesis space.

In general, problem (1) is substantially more challenging than its non-composition counterpart, i.e., empirical risk minimization (ERM) problem (Friedman, Hastie, and Tibshirani 2001) with the finite-sum form $f(x) = 1/n\sum_{i=1}^{n} F_i(x)$. On the one hand, the composition objective is nonlinear with respect to the joint distribution of data indices (i, j), which leads to the difficulty of an unbiased sampling for estimating the full gradient $\nabla f(x)$. On the other hand, the two finite-sums structure causes unprecedented computational challenges for traditional stochastic gradient methods in solving problem (1). For example, to apply stochastic gradient descent (SGD) method, we need to compute $(\partial G_j(x))^T \nabla F_i(G(x))$ in each iteration. The inner function $G(x)$ is computationally expensive with per-iteration queries proportional to n_2.

To solve problem (1), Wang et al. (Wang, Fang, and Liu 2014; Wang, Liu, and Tang 2017) first propose a class of stochastic compositional gradient descent methods, i.e., SCGD and ASC-PG, which are based on the random evaluations of $G(x)$, $F(y)$ and their gradients with a low per-iteration cost. However, SCGD and ASC-PG suffer from low convergence rate because of the variance induced by the random samplings. In recent years, variance reduction based methods have been proposed to improve the convergence rate for the composition optimization problem (1). Some algorithms employ stochastic variance reduction gradient (SVRG) method to improve SCGD, including Compositional-SVRG-1(Lian, Wang, and Liu 2017), Compositional-SVRG-2 (Lian, Wang, and Liu 2017), and

*Corresponding author

Table 1: Comparisons of query complexity for existing stochastic composition gradient algorithms including ASC-PG (Wang, Liu, and Tang 2017), Compositional-SVRG-1 (Lian, Wang, and Liu 2017), Compositional-SVRG-2 (Lian, Wang, and Liu 2017), com-SVR-ADMM (Yu and Huang 2017), VRSC-PG (Huo, Gu, and Huang 2018), and our proposed Async-ProxSCVR algorithm. n_1 denotes the number of outer component functions. n_2 denotes the number of inner component functions. κ is the condition number defined in section Convergence Analysis.

Method	$\mathbf{h(x) \neq 0}$	Asynchronous	Strongly convex problem	General nonconvex problem
ASC-PG	Yes	No	$O(1/\epsilon^{\frac{5}{4}})$	$O(1/\epsilon^{\frac{9}{4}})$
Compositional-SVRG-1	No	No	$O((n_1 + n_2 + \kappa^4)\log(1/\epsilon))$	-
Compositional-SVRG-2	No	No	$O((n_1 + n_2 + \kappa^3)\log(1/\epsilon))$	-
com-SVR-ADMM	Yes	No	$O((n_1 + n_2 + \kappa^4)\log(1/\epsilon))$	-
VRSC-PG	Yes	No	$O((n_1 + n_2 + \kappa^3)\log(1/\epsilon))$	$O((n_1 + n_2)^{\frac{2}{3}}/\epsilon)$
Async-ProxSCVR (Ours)	Yes	Yes	$O((n_1 + n_2 + \kappa^3)\log(1/\epsilon))$	$O((n_1 + n_2)^{\frac{2}{3}}/\epsilon)$

VRSC-PG (Huo, Gu, and Huang 2018). Compositional-SVRG-1 and Compositional-SVRG-2 are designed for problem (1) where $h(x) \equiv 0$, while VRSC-PG can deal with problem (1) with nonsmooth regularization term. In addition, (Yu and Huang 2017) proposes an algorithm, com-SVR-ADMM, which combines ideas of SVRG and ADMM for the composition optimization problem (1) with linear constraints. The query complexity[1] of these stochastic composition gradient algorithms are shown in Table 1. Although these existing methods for problem (1) may have good theoretical guarantees and analytical properties, they are inherently non-parallel (i.e., sequential) which are not scalable to modern large-scale composition optimization problems.

Another approach to improve the efficiency for solving problem (1) is the asynchronous parallel implementation, which has been successfully applied to speedup many state-of-the-art algorithms (Recht et al. 2011; Reddi et al. 2015; Zhang and Kwok 2014; Liu and Wright 2014; Liu et al. 2015). However, existing asynchronous parallel algorithms focus on solving the ERM problem. When directly utilizing existing asynchronous parallel methods, e.g., HOGWILD! (Recht et al. 2011) and Async-ProxSVRG (Meng et al. 2017) to solve problem (1), the inner function $G(x)$ and its Jacobi matrix $\partial G(x)$ still need to be computed in each iteration, which usually costs too much time and is inefficient for practical applications.

To fill this important gap, in this paper, we propose an asynchronous proximal variance reduction based algorithm, Async-ProxSCVR, for the composition optimization problem (1) with nonsmooth regularization term. We provide the convergence guarantees of the algorithm for both strongly convex and general nonconvex cases. We also prove that the total query complexity of our asynchronous algorithm would be comparable to its serial counterpart, which indicates a linear speedup. Our main contributions are listed as follows:

- By effectively combining the asynchronous parallel implementation with the variance reduction method, we develop an asynchronous proximal variance reduction method, Async-ProxSCVR, for problem (1) with nonsmooth regularization. Our method is time-efficient with low computational cost per iteration.
- We prove that Async-ProxSCVR achieves a query complexity of $O((n_1 + n_2 + \kappa^3)\log(1/\epsilon))$ for strongly convex case and $O((n_1 + n_2)^{\frac{2}{3}}/\epsilon)$ for general nonconvex case, which match the best results for existing stochastic composition algorithms. Furthermore, our analysis show that Async-ProxSCVR can achieve linear speedup with respect to the number of processors.
- We test our asynchronous parallel algorithm on two representative composition optimization problems including value function evaluation in reinforcement learning and sparse mean-variance optimization. The experimental results prove that our method achieves significant speedup and can converge faster than other existing stochastic composition gradient methods.

[1]The query complexity is measured in terms of the total number of component function queries to find an ϵ-accurate solution.

Notations

Throughout this paper, we use the following notations:

- $x^* \in \mathbb{R}^N$ denotes the optimal solution for problem (1).
- $||\cdot||$ denotes the Euclidean norm $||\cdot||_2$.
- $\partial G_j(x)$ denotes the Jacobi matrix of multivariate function $G_j(\cdot)$ at point x.
- $\nabla F_i(y)$ denotes the derivative function of multivariate function $F_i(\cdot)$ at point y.

Related Work

Asynchronous Parallel Algorithms. In recent years, asynchronous parallel algorithms have received broad attention and been successfully used in solving the ERM problem. Asynchronous parallel variants of many classical stochastic gradient algorithms have been proposed and analyzed in theory. Such as, stochastic gradient descent (SGD) (Recht et al. 2011), stochastic variance reduction gradient (SVRG) (Reddi et al. 2015), alternating direction method of multipliers (ADMM) (Zhang and Kwok 2014) and coordinate descent (Liu and Wright 2014; Liu et al. 2015). Inspired by these recent theoretical advances and successful applications in the ERM problem, we study the asynchronous parallel stochastic gradient method for problem (1) with nonsmooth regularization penalty. The closest work to ours is the Async-ProxSVRG (Fang and Lin 2017) designed for the regularized ERM problem. However, to solve problem (1), the Async-ProxSVRG requires high $O(n_2)$ queries per iteration which are computationally expensive.

Variance Reduction Methods. To alleviate the adverse effect of randomness inherent in stochastic gradient algorithms, several variance reduction methods have been proposed, e.g., SVRG (Johnson and Zhang 2013), SAG (Schmidt, Le Roux, and Bach 2017), SDCA (Shalev-Shwartz and Zhang 2013) and SAGA (Defazio, Bach, and Lacoste-Julien 2014). Due to its low memory overhead, we focus on the SVRG method, which is initially used to solve the ERM problem with the finite-sum form $f(x) = 1/n \sum_{i=1}^{n} F_i(x)$. The SVRG algorithm has double loops. More specifically, at the s^{th} epoch, the SVRG keeps a snapshot of iteration vector $\widetilde{x}^s$ and compute its full gradient $\nabla f(\widetilde{x}^s)$. In the k^{th} inner loop, the SVRG compute gradient estimator based on the following update rule:

$$v_k^s = \nabla F_{i_k}(x_k^s) - \nabla F_{i_k}(\widetilde{x}^s) + \nabla f(\widetilde{x}^s), \tag{3}$$

where i_k is randomly selected from $[n]$. This update rule has a lower iteration cost, because there is no need to calculate the full gradient in each iteration. The gradient estimator is unbiased, i.e., $\mathbb{E}[v_k^s] = \nabla f(x_k^s)$. Besides, the variance of the gradient estimator induced by the random sampling is upper bounded (Johnson and Zhang 2013):

$$\mathbb{E}[||v_k^s||^2] \leqslant 4L_f[f(x_k^s) - f(x^*) + f(\widetilde{x}^s) - f(x^*)]. \tag{4}$$

When x_k^s and $\widetilde{x}$ approach the optimal solution x^*, the variance of gradient goes to 0. Furthermore, The SVRG method has a provably faster convergence rate, i.e., $O(n \log \frac{1}{\epsilon})$ for strongly convex case (Johnson and Zhang 2013) and $O(\frac{n^{\frac{2}{3}}}{\epsilon})$ for nonconvex case (Reddi et al. 2016). Recently, many variants of the SVRG method have achieved great success, e.g., the proximal SVRG method (Xiao and Zhang 2014), the asynchronous SVRG method (Reddi et al. 2015) and the distributed SVRG method (Lee et al. 2015). In our method, we introduce a SVRG-based compositional variance reduction method to effectively solve the composition problem (1).

Asynchronous Proximal Stochastic Compositional Variance Reduction Algorithm

In this section, we propose Async-ProxSCVR to solve the composition optimization problem (1) with nonsmooth regularization penalty. Our proposed Async-ProxSCVR exploits the asynchronous parallel scheme and compositional variance reduction method, which is significantly different from existing works (Fang and Lin 2017; Huo, Gu, and Huang 2018; Yu and Huang 2017) as discussed before.

1. For asynchronous parallel implementation, we use the shared memory multi-core computational model that multiple processors have free access to the vector x stored in the shared memory and independently execute the iterative task concurrently without using any locks[2].

[2]This may result in the *inconsistent reading and writing*, which makes the convergence analysis of the proposed Async-ProxSCVR method more challenging.

Algorithm 1: Async-ProxSCVR

Input: initial vector $\widetilde{x}^0$, step size η, number of inner loops m, number of outer loops S, size of mini-batch A, B and I.
Output: $\widetilde{x}^S$ and x_a, where x_a is uniformly selected from $\{\{x_k^s\}_{k=0}^{m-1}\}_{s=0}^{S-1}$.
for $s = 0, 1, \cdots, S-1$ **do**
  $x_0^s = \widetilde{x}^s$;
  `// ` n_2 `queries`
  $G(\widetilde{x}^s) = \frac{1}{n_2} \sum_{j=1}^{n_2} G_j(\widetilde{x}^s)$;
  `// ` n_2 `queries`
  $\partial G(\widetilde{x}^s) = \frac{1}{n_2} \sum_{j=1}^{n_2} \partial G_j(\widetilde{x}^s)$;
  `// ` n_1 `queries`
  $\nabla f(\widetilde{x}^s) = (\partial G(\widetilde{x}^s))^T \frac{1}{n_1} \sum_{i=1}^{n_1} \nabla F_i(G(\widetilde{x}^s))$;
  Parallel Computation on Multiple Threads
  for $k = 0, 1, \cdots, m-1$ **do**
    Read $x_{D(k)}^s$ from the shared memory;
    Randomly choose $A_k \subset [n_2]$ with $|A_k| = A$;
    `// ` $2A$ `queries`
    Compute $\hat{G}_{D(k)}^s$ from Eq.(6);
    Randomly choose $B_k \subset [n_2]$ with $|B_k| = B$;
    `// ` $2B$ `queries`
    Compute $\partial \hat{G}_{D(k)}^s$ from Eq.(7);
    Randomly choose $I_k \subset [n_1]$ with $|I_k| = I$;
    `// ` $2I$ `queries`
    Compute v_k^s from Eq.(8);
    Update $x_{k+1}^s = \text{Prox}_{\eta h}(x_k^s - \eta v_k^s)$;
  end
  Option I: $\widetilde{x}^{s+1} = x_m^s$;
  Option II: $\widetilde{x}^{s+1} = \frac{1}{m} \sum_{k=1}^{m} x_k^s$;
end

---

2. For compositional variance reduction method, considering the two finite-sums structure in problem (1), our proposed Async-ProxSCVR employs variance reduction method to estimate the inner function $G(\cdot)$, the partial gradient of inner function $\partial G(\cdot)$, and the gradient of full function $\nabla f(\cdot)$ with low per-iteration queries.

More specifically, following the SVRG method (Johnson and Zhang 2013), the proposed Async-ProxSCVR algorithm has two-layer loops. At the beginning of outer epoch s, we compute the full gradient $\nabla f(\widetilde{x}^s)$ at the snapshot point $\widetilde{x}^s$:

$$\nabla f(\widetilde{x}^s) = (\partial G(\widetilde{x}^s))^T \nabla F(G(\widetilde{x}^s)), \tag{5}$$

where $G(\widetilde{x}^s) = \frac{1}{n_2} \sum_{j=1}^{n_2} G_j(\widetilde{x}^s)$ denotes the value of the inner function and $\partial G(\widetilde{x}^s) = \frac{1}{n_2} \sum_{j=1}^{n_2} \partial G_j(\widetilde{x}^s)$ denotes the full gradient of inner function. Computing the full gradient $\nabla f(\widetilde{x}^s)$ requires a total of $(n_1 + 2n_2)$ queries.

In the inner loop, each processor repeatedly runs the following steps: *read* the iteration vector from the shared memory, *compute* the gradient estimator, and then *update* the iteration vector stored in the shared memory. Specifically, at the k-th iteration of s-th outer epoch, with $A_k \subset [n_2]$,

$B_k \subset [n_2]$ being random sampling sets, $|A_k| = A$ and $|B_k| = B$, we compute the estimated values of $G(\cdot)$ and $\partial G(\cdot)$ at the iteration vector $x^s_{D(k)}$ read from the shared memory:

$$\hat{G}^s_{D(k)} := \frac{1}{A} \sum_{1 \leqslant j \in A} (G_{A_{k[j]}}(x^s_{D(k)}) - G_{A_{k[j]}}(\widetilde{x}^s)) + G(\widetilde{x}^s), \tag{6}$$

$$\partial \hat{G}^s_{D(k)} := \frac{1}{B} \sum_{1 \leqslant j \leqslant B} (\partial G_{B_{k[j]}}(x^s_{D(k)}) - \partial G_{B_{k[j]}}(\widetilde{x}^s)) + \partial G(\widetilde{x}^s), \tag{7}$$

where A_k and B_k are independent. Note that the query complexities of computing $\hat{G}^s_{D(k)}$ and $\partial \hat{G}^s_{D(k)}$ are independent of the number of inner component functions, i.e., n_2.

Now, we compute the gradient estimator as follows:

$$v^s_k := \frac{1}{I} \sum_{i_k \in I_k} ((\partial \hat{G}^s_{D(k)})^T \nabla F_{i_k}(\hat{G}^s_{D(k)}) - (\partial G(\widetilde{x}^s))^T \nabla F_{i_k}(G(\widetilde{x}^s))) + \nabla f(\widetilde{x}^s), \tag{8}$$

where $I_k \subset [n_1]$ is random sampling set with $|I_k| = I$. Finally, we update the iteration vector x^s_{t+1} through a proximal operator:

$$x^s_{k+1} = \text{Prox}_{\eta h}(x^s_k - \eta v^s_k), \tag{9}$$

where η is the learning rate. For each inner loop, computing the gradient estimator requires $2(A + B + I)$ qeuries. After m inner iterations, we update the snapshot point $\widetilde{x}^{s+1}$ based on Option I or Option II for the next epoch. The detailed Async-ProxSCVR algorithm is presented in Algorithm 1.

In the inner loop, we maintain the iteration counter k to track the number of *update* operations. Since update happens asynchronously, there will generally be a delay between the current iteration vector and the iteration vector used to calculate the gradient estimator. In Eq.(6), Eq.(7), and Eq.(8), we use $D(k)$ to denote the iteration counter corresponding to the time that the iteration vector in the shared memory is read at the k^{th} iteration. The iteration value $x^s_{D(k)}$ can be represented as below:

$$x^s_{D(k)} = x^s_{k-\tau_k} + \sum_{h \in J(k)} (x_{h+1} - x_h), \tag{10}$$

where $J(k) \subset \{k - \tau_k, \cdots, k - 1\}$, and τ_k denotes the time delay of the gradient estimator at the k^{th} iteration.

Note that the gradient estimator v^s_k in Algorithm 1 is a biased estimation of $\nabla f(x^s_{D(k)})$ significantly different from the unbiased estimation in standard SVRG method. Hence, the convergence analysis of Async-ProxSCVR is more complex than existing asynchronous variance reduction algorithms (Reddi et al. 2015; Meng et al. 2017).

Convergence Analysis

We now proceed to analyze the convergence behavior and evaluate the query complexity of our proposed Async-ProxSCVR for both strongly convex and general nonconvex cases. We first make some commonly used assumptions in the literatures of composition optimization problems (Lian, Wang, and Liu 2017; Huo, Gu, and Huang 2018; Yu and Huang 2017).

Assumption 1 (Lipschitzian Gradients)**.** *For $i \in [n_1]$, $j \in [n_2]$, there exist constants $0 < L_F, L_G, L_f < \infty$ such that*

$$\begin{aligned}
&||\nabla F_i(y_1) - \nabla F_i(y_2)|| \leqslant L_F||y_1 - y_2||, \forall y_1, y_2 \in \mathbb{R}^M,\\
&||\partial G_j(x_1) - \partial G_j(x_2)|| \leqslant L_G||x_1 - x_2||, \forall x_1, x_2 \in \mathbb{R}^N,\\
&||(\partial G_j(x_1))^T \nabla F_i(G(x_1)) - (\partial G_j(x_2))^T \nabla F_i(G(x_2))||\\
&\leqslant L_f||x_1 - x_2||, \forall x_1, x_2 \in \mathbb{R}^N.
\end{aligned}$$

Assumption 2 (Bounded Gradients)**.** *For $i \in [n_1]$, $j \in [n_2]$, there exist some constants $0 < B_F, B_G < \infty$ such that*

$$||\nabla F_i(y)|| \leqslant B_F, ||\partial G_j(x)|| \leqslant B_G, \forall x \in \mathbb{R}^N, \forall y \in \mathbb{R}^M.$$

Assumption 3 (Bounded Delay)**.** *There exists a delay parameter τ such that the random delay variables τ_k are upper bounded by τ, i.e., $\tau_k \leqslant \tau$ in Async-ProxSCVR.*

Let $\sigma := B_G^4 L_F^2/A + B_F^2 L_G^2/B$. According to Assumption 1, we can immediately arrive at a corollary that $f(x)$ is L_f-smooth function, i.e.,

$$||\nabla f(x_1) - \nabla f(x_2)|| \leqslant L_f||x_1 - x_2||, \forall x_1, x_2 \in \mathbb{R}^N.$$

Convergence Analysis in Strongly Convex Case

For the strongly convex case, we prove that our Async-ProxSCVR admits linear convergence rate, and the query complexity to achieve ϵ-accurate solution is $O((n_1 + n_2 + \kappa^3)\log(1/\epsilon))$. As in (Reddi et al. 2015; Recht et al. 2011; Meng et al. 2017), we consider our algorithm in the sparse setting that $F_i(y)$ acts only some components of y. We introduce the sparse constant Δ into our convergence analysis.

Assumption 4. *The objective $f(x)$ and $h(x)$ are both convex. Furthermore, the objective $H(x)$ is μ-strongly convex, i.e., $\forall x_1, x_2 \in \mathbb{R}^N$ and $\forall \xi \in \partial H(x_1)$, we have:*

$$H(x_2) \geqslant H(x_1) + \xi^T(x_2 - x_1) + \frac{\mu}{2}||x_2 - x_1||^2.$$

Theorem 1. *Assume that Assumptions 1-4 hold. If the step size $\eta \leqslant \min\{\frac{\mu}{128\sigma\tau^2}, \frac{1}{4L_f I \Delta \tau^2}, \frac{1}{6L_f}\}$, the mini-batch sizes A, B, I and the inner loop size m are selected properly so that*

$$\rho := \frac{\frac{2}{\mu} + 2\eta(\frac{12\eta L_f}{I} + (\eta + \frac{4}{\mu})\frac{16\sigma}{\mu})(m+1)}{2\eta\left(\frac{7}{8} - (\frac{12\eta L_f}{I} + (\eta + \frac{4}{\mu})\frac{16\sigma}{\mu})\right)m} < 1, \tag{11}$$

then the Async-ProxSCVR with Option II has linear convergence rate in expectation:

$$\mathbb{E}[H(\widetilde{x}^s) - H(x^*)] \leqslant \rho^s[H(\widetilde{x}^0) - H(x^*)]. \tag{12}$$

Proof. See Section C.1 in the supplementary material. □

Remark 1. *We now emphasize that there exist values of the parameters m, A, B and I for which the inequality (11) in Theorem 1 are easily satisfied. For instance, setting the* *mini-batch sizes* $A = \max\{\frac{1024B_G^4L_F^2}{\mu^2}, \frac{256\eta B_G^4L_F^2}{\mu}\}$ *and* $B = \max\{\frac{1024B_F^2L_G^2}{\mu^2}, \frac{256\eta B_F^2L_G^2}{\mu}\}$*, we then have* $(\eta + \frac{4}{\mu})\frac{16\sigma}{\mu} \leqslant 1/4$*. If we set* $\eta = \frac{1}{6L_f}$ *and* I *is sufficiently large (s.t.* $\frac{1}{6L_f} \leqslant \frac{I}{192L_f}$*), we have* $\frac{12\eta L_f}{I} \leqslant \frac{1}{16}$*, and then* $\rho \leqslant \frac{5}{9} + \frac{\frac{16}{\mu m}+\frac{5\eta}{m}}{9\eta}$*. With the number of inner loop* $m = 16(1+\frac{1}{\mu\eta})$*, we have* $\rho \leqslant \frac{2}{3}$*. According to Theorem 1 and the aforementioned discussions, we provide the following corollary.*

Corollary 1. *According to Theorem 1, if the delay parameter satisfies* $\tau^2 \leqslant \min\{\frac{3}{2I\Delta}, \frac{3\mu L_f}{64\sigma}\}$*, we set* $\eta = \frac{1}{6L_f}$*,* $m = 16(1+\frac{6L_f}{\mu})$*,* $A = \frac{1024B_G^4L_F^2}{\mu^2}$*,*$B = \frac{1024B_F^2L_G^2}{\mu^2}$*, I is sufficiently large (s.t.* $\frac{1}{6L_f} \leqslant \frac{I}{192L_f}$*), the Async-ProxSCVR with Option II has the following linear convergence rate:*

$$\mathbb{E}[H(\widetilde{x}^s) - H(x^*)] \leqslant (\frac{2}{3})^s[H(\widetilde{x}^0) - H(x^*)]. \quad (13)$$

Remark 2. *Corollary 1 presents an example to show how to choose* m*,* A*,* B *and* I *appropriately. It can be observed that the values of our selected parameters* m*,* A*,* B *and* I *for Algorithm 1 are comparable to ones of its sequential counterpart (Huo, Gu, and Huang 2018). According to the definition of Sampling Oracle in (Wang, Liu, and Tang 2017), to make* $\mathbb{E}[H(\widetilde{x}^s) - H(x^*)] \leqslant \epsilon$*, the total query complexity is* $O\left((n_1+n_2+m(A+B+I))\log(\frac{1}{\epsilon})\right)$*. We denote* $\kappa := \max\{\frac{L_F}{\mu}, \frac{L_G}{\mu}, \frac{L_f}{\mu}\}$ *and set the parameters* $m = O(\kappa)$*,* $A = O(\kappa^2)$*,* $B = O(\kappa^2)$*,* $I = O(1)$ *as in Corollary 1. The total query complexity is* $O((n_1+n_2+\kappa^3)\log(\frac{1}{\epsilon}))$ *which is the same as the optimal sequential algorithm VRSC-PG (Huo, Gu, and Huang 2018). From inequality (13), we know that the convergence rate has nothing to do with the delay parameter if it is upped bounded by* $\min\{\sqrt{\frac{3}{2I\Delta}}, \sqrt{\frac{3\mu L_f}{64\sigma}}\}$*, and then linear speedup is accessible under this condition.*

Convergence Analysis in General Nonconvex Case

In this subsection, we consider the proposed Async-ProxSCVR in general nonconvex case. We remove the sparse assumption that appears in the strongly convex case. For general nonconvex problems, there is no guarantee that the optimal solution x^* is finally reached. Consequently, we employ the alternative evaluation metric $\mathcal{G}_\eta(x) := \frac{1}{\eta}(x - \text{Prox}_{\eta h}(x - \nabla f(x)))$, which is common for general nonconvex problem (J. Reddi et al. 2016; Huo and Huang 2017).

Theorem 2. *Assume that Assumptions 1-3 hold. If the step size* $\eta \leqslant \frac{1}{\sqrt{12\sigma\tau}}$*, the mini-batch sizes A, B, I, and the inner loop size* m *are selected properly such that:*

$$\frac{L_f}{2} + 12\eta m^2(\frac{B_G^4L_F^2}{A} + \frac{B_F^2L_G^2}{B} + \frac{L_f^2}{2I}) \leqslant \frac{1}{4\eta}, \quad (14)$$

then the Async-ProxSCVR with Option I has a sublinear convergence rate in expectation:

$$\mathbb{E}[||\mathcal{G}_\eta(x_a)||^2] \leqslant \frac{2\eta}{(1-2\eta L_f)}\frac{H(\widetilde{x}^0) - H(x^*)}{T}, \quad (15)$$

where x_a *is uniformly selected from* $\{\{x_k^{s+1}\}_{k=0}^{m-1}\}_{s=0}^{S-1}$*, and T be a multiple of* m*.*

Proof. See Section C.2 in the supplementary material. □

We now provide an example to show how to select these parameters A, B, I and m to satisfy the inequality (14).

Corollary 2. *If the delay parameter* $\tau \leqslant \sqrt{12L_f^2/\sigma}$*, we set* $\eta = \frac{1}{12L_f}$*, the mini-batch sizes* $A = \frac{2m^2B_G^4L_F^2}{L_f^2}$*,* $B = \frac{2m^2B_F^2L_G^2}{L_f^2}$*,* $I = (n_1+n_2)^{\frac{2}{3}}$ *and* $m = \lfloor (n_1+n_2)^{\frac{1}{3}} \rfloor$*, we have the inequality (14) holds. Then, the Async-ProxSCVR with Option I has the following sublinear convergence rate:*

$$\mathbb{E}[||\mathcal{G}_\eta(x_a)||^2] \leqslant \frac{1}{5L_f}\frac{H(\widetilde{x}^0) - H(x^*)}{T}. \quad (16)$$

Remark 3. *Corollary 2 presents an example to show how to choose* m*,* A*,* B *and* I *appropriately. To achieve the* ϵ*-accurate solution, i.e.,* $\mathbb{E}[||\mathcal{G}_\eta(x_a)||^2] \leqslant \epsilon$*, the query complexity we need to take is* $O((n_1+n_2+m(A+B+I))\frac{1}{m\epsilon})$*. With the same parameters as in Corollary 2, we set* $m = O((n_1+n_2)^{\frac{1}{3}})$*,* $A = O((n_1+n_2)^{\frac{2}{3}})$*,* $B = O((n_1+n_2)^{\frac{2}{3}})$*,* $I = O((n_1+n_2)^{\frac{2}{3}})$*, the total query complexity becomes* $O((n_1+n_2)^{\frac{2}{3}}/\epsilon)$*, which is the same as the optimal sequential algorithm VRSC-PG (Huo, Gu, and Huang 2018). In inequality (16), we find out that the convergence rate has nothing to do with the delay parameter* τ*, if it is upper bounded by* $\sqrt{12L_f^2/\sigma}$*. Thus in this specific domain, a linear speedup is accessible when we increase the number of processors.*

Experiments

In this section, we conduct numerical experiments to verify the convergence guarantees and the linear speedup of the proposed Async-ProxSCVR algorithm in a shared memory multi-core system. In our experiments, we involve two representative composition optimization problems including value function evaluation in reinforcement learning (RL) and sparse mean-variance optimization problem.

Application to Value Function Evaluation in RL

Almost all reinforcement learning algorithms involve value function estimation. It is known that value function satisfies the Bellman equation (Sutton and Barto 1998), that is,

$$v_\pi(s_1) = \mathbb{E}_\pi[r_{s_1,s_2} + \gamma \cdot v_\pi(s_2)|s_1], \quad (17)$$

where $s_1, s_2 \in \{1, 2, \cdots, S\}$, π is the policy and $v_\pi(s)$ denotes the value function that starting from state s and following policy π thereafter. To solve the linear value function evaluation through the Bellman Equation, the objective function can be formulated as follows:

$$w^* = \arg\min_{x\in\mathbb{R}^d} \sum_{s=1}^{S}(w^T\varphi(s) - q_{\pi,s}(w))^2 + \lambda||w||_1, \quad (18)$$

where $w^T\varphi(s)$ is the linear approximation to $v_\pi(s)$, $\varphi(\cdot) \in \mathbb{R}^d$ is a pre-defined feature map of state s, and $q_{\pi,s}(w) = \mathbb{E}_\pi[r_{s,s'} + \gamma \cdot w^T\varphi(s')] = \sum_{s'} P_{s,s'}^\pi(r_{s,s'} + \gamma \cdot w^T\varphi(s'))$. $P_{s,s'}^\pi$ is the transition probability from state s to s' under

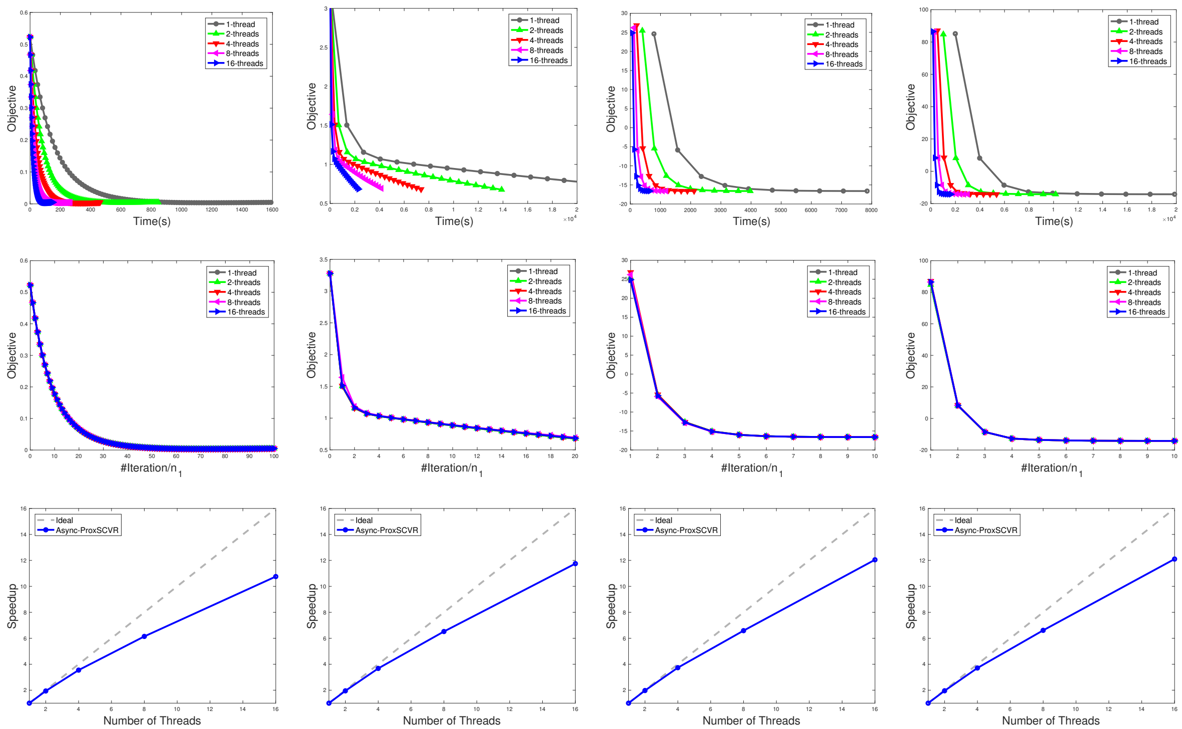

Figure 1: Results of the speedup experiment. Columns, from left to right, are the results on MDPs (100 states, 3 actions), MDPs (400 states, 10 actions), mean-variance problem (N=2000,d=300) and mean-variance problem (N=5000,d=300). The top row shows the curve of objective w.r.t the running time. The middle row shows the curve of objective w.r.t the number of iterations. The bottom row represents the time speedup with varying threads, where gray line represents ideal linear speedup.

policy π, $r_{s,s'}$ is the immediate reward from s to s', and $||w||_1$ is the regularization term for sparsity. As mentioned in (Wang, Liu, and Tang 2017), the objective function can be reformulated as the composition optimization problem (1) in which:

$$G(w) = \left(w^T\varphi(1), q_{\pi 1}(w), \cdots, w^T\varphi(S), q_{\pi S}(w)\right), \quad (19)$$

$$F((x_1, y_1, \cdots, x_S, y_S)) = \sum_{s=1}^{S}(x_s - y_s)^2. \quad (20)$$

Following (Wang, Liu, and Tang 2017; Dann, Neumann, and Peters 2014), our experiments are conducted on random MDPs. In our first experiment, the MDP instance contains 100 states, and 3 actions at each state. The transition probabilities and rewards for each transition are uniformly sampling form [0, 1]. The sum of transition probabilities is then normalized to one. In our second experiment, the MDP instance contains 400 states, and 10 actions at each state. The transition probabilities and rewards are similarly sampling as in the first experiment. For both two experiments, we set regularization parameter $\gamma = 10^{-5}$, the discounted factor $\lambda = 0.95$. These are well-known examples to test the off-policy convergent algorithms (Geist and Scherrer 2014).

Application to Sparse Mean-variance Optimization

For the high-dimensional sparse portfolio management problem, the goal is to create a portfolio that have maximum expected return but minimal variance of return. Specifically, given d assets, let $\{r_i\}_{i=1}^N \subset \mathbb{R}^d$ be the reward vectors at different time points, the objective function can be formulated as the following sparse mean-variance optimization:

$$x^* = \arg\min_{x\in\mathbb{R}^d}\Big[\frac{1}{N}\sum_{i=1}^{N}(\langle r_i, x\rangle - \frac{1}{N}\sum_{j=1}^{N}\langle r_j, x\rangle)^2 - \frac{1}{N}\sum_{i=1}^{N}\langle r_i, x\rangle + \lambda||x||_1\Big], \quad (21)$$

where $x \in \mathbb{R}^d$ denotes the investment quantity in d assets. Following (Huo, Gu, and Huang 2018; Yu and Huang 2017), the problem (21) is in the form of problem (1) in which:

$$G_j(x) = (x, -\langle r_j, x\rangle)^T, \quad x \in \mathbb{R}^d, \quad (22)$$

$$F_i((x, y)) = (\langle r_i, x\rangle + y)^2 - \langle r_i, x\rangle, x \in \mathbb{R}^d, y \in \mathbb{R}. \quad (23)$$

In the experiment, we set $N \in \{2000, 5000\}$, $d = 300$, $m = 30$ and the regularization parameter $\gamma = 10^{-5}$. The reward vectors are generated through Gaussian distributions with zero mean and positive semi-definite co-variance matrix $\Sigma = L^T L$, where $L \in \mathbb{R}^{d\times m}$ and $m < d$. Each element of L is drawn from the normal distribution.

Compared Methods. In our experiments, there are five compared methods. More specifically, **HOGWILD!** (Recht

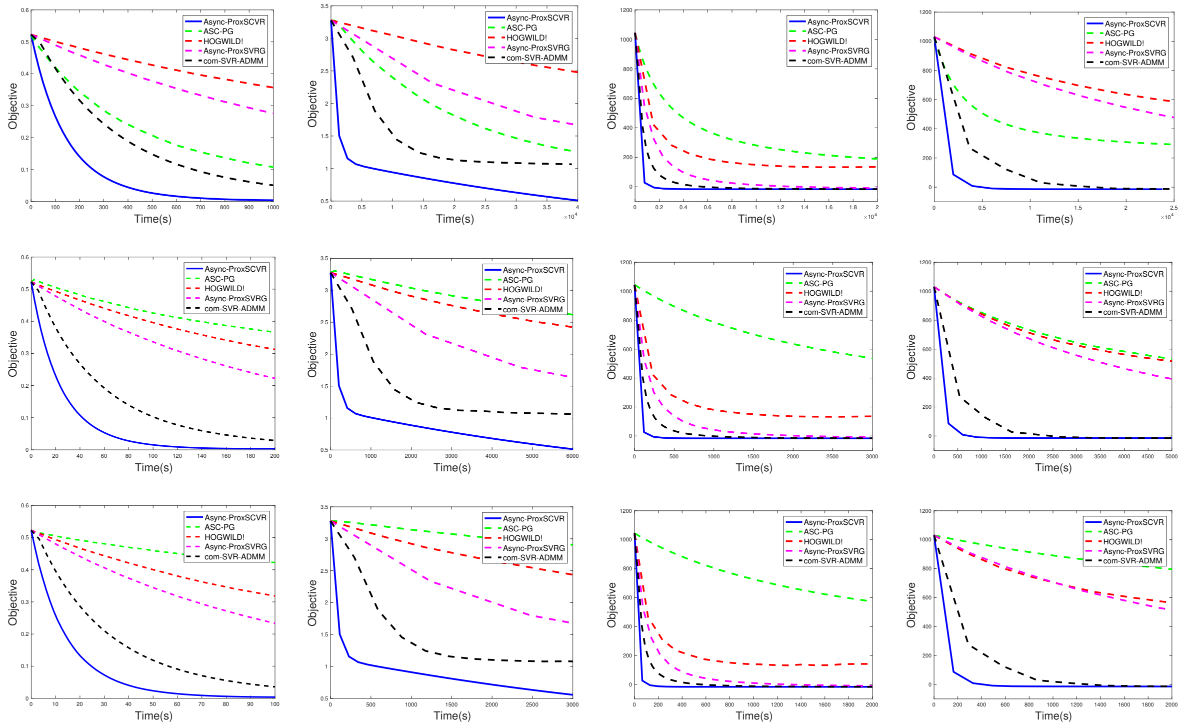

Figure 2: Comparison of five compared methods: HOGWILD!, Async-ProxSVRG, ASC-PG, com-SVR-ADMM and our proposed Async-ProxSCVR. As in Figure 1, columns, from left to right, are the results on MDPs (100 states, 3 actions), MDPs (400 states, 10 actions), mean-variance problem (N=2000,d=300) and mean-variance problem (N=5000,d=300). The rows from top to bottom represent the results on 1, 8, and 16 threads, respectively.

et al. 2011), **Async-ProxSVRG** (Meng et al. 2017), **ASC-PG** (Wang, Liu, and Tang 2017), **com-SVR-ADMM** (Yu and Huang 2017) and the proposed **Async-ProxSCVR**.

Experimental Setting. For a fair comparison, we use the best tuned learning rates for compared methods, specifically, we select them from $\{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. For variance reduction based algorithms, we set the inner loop size $m = kn_1$, in which k is best tuned from $\{0.5, 1, 2\}$. The mini-batch sizes of A, B and I are chosen casually as in (Huo, Gu, and Huang 2018). Furthermore, all of the methods are coded in C++ using the standard thread library. We conduct all the experiments on an identical server which has 16 Intel(R) Xeon(R) E5-2650 CPUs and 64GB memory.

Experimental Results

Scaling with Number of Threads. In Figure 1, we show the speedup results of Async-ProxSCVR with varying number of threads. The top line are results of objective function value with respect to running time. We observe that when we use more threads, the objective curve decreases faster. The middle line represents results of objective function value with respect to the number of iterations. The figures show that objective curves with different threads are almost coincident. It means that the negative effect of using stale iteration vector for calculating the gradient estimator vanishes to some extent. The bottom line shows results of the running time speedup. We can observe that the algorithm has a strong scaling when we use multiple threads. For instance, our proposed algorithm can achieve about $12\times$ speedup on 16 threads. Besides, results show that when the number of threads increases, our method has a nearly linear speedup in all experiments. These findings in the experiment are compatible with our theoretical analysis.

Comparison with Other Methods. To demonstrate the superiority of our proposed Async-ProxSCVR, we conduct comparison experiments with all baseline methods. Figure 2 presents the objective curves versus the running time on 1, 8 and 16 threads, respectively. Results show that our proposed algorithm Async-ProxSCVR achieves fast empirical convergence rate compared with other benchmark algorithms in all four experiments. Furthermore, we see that our proposed algorithm is more robust to increasing threads.

Conclusion

In this paper, we introduce an asynchronous parallel method Async-ProxSCVR for problem (1) with nonsmooth regularization penalty. We prove that our proposed Async-ProxSCVR can achieve a linear convergence rate $O((n_1 + n_2 + \kappa^3)\log(1/\epsilon))$ for strongly convex case and a sublinear convergence rate $O((n_1 + n_2)^{\frac{2}{3}}/\epsilon)$ for general nonconvex case. Furthermore, linear speedup is accessible when the delay parameter is upper bounded. Numerical experiments on two representative composition optimization problems are conducted to demonstrate our convergence analysis.

Acknowledgments

This work is supported by the Zhejiang Provincial Natural Science Foundation (LR19F020005), National Natural Science Foundation of China (61572433, 61672125, 31471063, 61473259, 31671074) and thanks for a gift grant from Baidu inc. Also partially supported by the Hunan Provincial Science & Technology Project Foundation (2018TP1018, 2018RS3065) and the Fundamental Research Funds for the Central Universities.

References

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

Defazio, A.; Bach, F.; and Lacoste-Julien, S. 2014. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, 1646–1654.

Fang, C., and Lin, Z. 2017. Parallel asynchronous stochastic variance reduction for nonconvex optimization. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, 794–800.

Friedman, J.; Hastie, T.; and Tibshirani, R. 2001. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:.

Geist, M., and Scherrer, B. 2014. Off-policy learning with eligibility traces: A survey. *The Journal of Machine Learning Research* 15(1):289–333.

Hinton, G. E., and Roweis, S. T. 2003. Stochastic neighbor embedding. In *Advances in neural information processing systems*, 857–864.

Huo, Z., and Huang, H. 2017. Asynchronous mini-batch gradient descent with variance reduction for non-convex optimization. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*.

Huo, Z.; Gu, B.; and Huang, H. 2018. Accelerated method for stochastic composition optimization with nonsmooth regularization. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*.

J. Reddi, S.; Sra, S.; Poczos, B.; and Smola, A. J. 2016. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *Advances in neural information processing systems*, 1145–1153.

Johnson, R., and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, 315–323.

Lee, J. D.; Lin, Q.; Ma, T.; and Yang, T. 2015. Distributed stochastic variance reduced gradient methods and a lower bound for communication complexity. *arXiv preprint arXiv:1507.07595*.

Lian, X.; Wang, M.; and Liu, J. 2017. Finite-sum composition optimization via variance reduced gradient descent. In *The 20th International Conference on Artificial Intelligence and Statistics*.

Liu, J., and Wright, S. J. 2014. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization* 25(1).

Liu, J.; Wright, S. J.; Ré, C.; Bittorf, V.; and Sridhar, S. 2015. An asynchronous parallel stochastic coordinate descent algorithm. *The Journal of Machine Learning Research* 16(1):285–322.

Meng, Q.; Chen, W.; Yu, J.; Wang, T.; Ma, Z.; and Liu, T.-Y. 2017. Asynchronous stochastic proximal optimization algorithms with variance reduction. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*.

Recht, B.; Re, C.; Wright, S.; and Niu, F. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, 693–701.

Reddi, S. J.; Hefny, A.; Sra, S.; Poczos, B.; and Smola, A. J. 2015. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in neural information processing systems*, 2647–2655.

Reddi, S. J.; Hefny, A.; Sra, S.; Poczos, B.; and Smola, A. 2016. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, 314–323.

Schmidt, M.; Le Roux, N.; and Bach, F. 2017. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming* 162(1-2):83–112.

Shalev-Shwartz, S., and Zhang, T. 2013. Stochastic dual coordinate ascent methods for regularized loss minimization. *The Journal of Machine Learning Research* 14(2):567–599.

Shapiro, A.; Dentcheva, D.; and Ruszczyński, A. 2009. *Lectures on stochastic programming: modeling and theory*. SIAM.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Wang, M.; Fang, E. X.; and Liu, H. 2014. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *arXiv preprint arXiv:1411.3803*.

Wang, M.; Liu, J.; and Tang, E. X. 2017. Accelerating stochastic composition optimization. *Journal of Machine Learning Research* 18.

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

Yu, Y., and Huang, L. 2017. Fast stochastic variance reduced admm for stochastic composition optimization. *arXiv preprint arXiv:1705.04138*.

Zhang, R., and Kwok, J. 2014. Asynchronous distributed admm for consensus optimization. In *International conference on machine learning*, 1701–1709.