Journal of Network Intelligence
Taiwan Ubiquitous Information
©2025 ISSN 2414-8105 (Online)
Volume **10**, Number **2**, May **2025**

Asynchronous Parallel Stochastic Proximal Gradient Methods for Non-Convex Composite Optimization

Wen-Wu He

School of Computer Science and Mathematics
Fujian University of Technology, Fuzhou 350118, China
Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fuzhou 350118, China
hwwhbb@163.com

Xuan-Jin Gong

School of Computer Science and Mathematics
Fujian University of Technology, Fuzhou 350118, China
gxjmars@163.com

Zhuo-Xu Cui*

School of Mathematics and Statistics,
Wuhan University, Wuhan 430072, China
zx.cui@siat.ac.cn

*Corresponding author: Zhuo-Xu Cui

Received January 14, 2024, revised May 18, 2024, accepted October 8, 2024.

Abstract. *This paper considers the parallel stochastic composite optimization for general asynchronous cases where both the objective function and the regularizer are non-convex. In particular, based on the randomized stochastic proximal gradient protocol, an asynchronous parallel algorithm termed ARSPG is proposed which incorporates well the non-convex regularization to obtain the desired oracle property. Theoretical analysis shows that ARSPG has the convergence rate in the order of $\mathcal{O}(1/\epsilon^2)$. Further, a gradient-free version termed ARSPGF is developed to handle cases where only noisy objective function values are available. The analysis shows that ARSPGF has the same order of convergence rate as that of the normal version. Experiments over benchmark data sets validate well theoretical analyses and experimental results also show interesting practical aspects of the proposed algorithms.*
Keywords: Non-convex Composite Optimization, Parallel Computation, Asynchronous Algorithms, Stochastic Proximal Gradient Methods, Large-Scale Optimization.

1. **Introduction.** In recent years, there have been growing interests on stochastic optimization, e.g., Stochastic Gradient Descent (SGD) methods [1, 2, 3, 4, 5, 6]. Owing to its cheaper cost in each iteration, compared with classic Gradient Descent (GD), SGD becomes one of the most popular first-order methods to solve optimization problems in large-scale [7, 8, 9]. In practice, it often happens that one single computer cannot process the huge amounts of data and studies on asynchronous parallel optimization have received broad attentions, of which, the key idea is to allow all processors work independently and without the need of synchronization or coordination [10, 11]. With the asynchronism, the idle time can be reduced and processors with fast working paces can provide more updates. As has been consistently reported in [12], [13] and [14], thanks to the asynchrony, the computation time can be decreased almost linearly with the number of processors.

Consequently, it becomes a hot topic to combine the asynchronous parallelism with SGD [6, 11, 15, 16, 17].

On the other hand, the successful development of non-convex models, such as methods on regularization with sparsity [18, 19, 20, 21] or deep neural networks [22, 23, 24, 25], motivates the efforts on efficient methods for non-convex optimization. In general, it is NP-hard to find a global minimum for non-convex functions [26]. Instead, more practicable ideas are often proposed to find local minima. Accordingly, it is of great interest to develop SGD methods based on asynchronous parallelism for non-convex optimization in large-scale, e.g., to train neural networks [11, 27]. There are interesting methods based on evolutionary computation such as [28, 29, 30], which also provide insightful solutions to complex optimization problems.

In this paper, we focus on the stochastic composite optimization (SCO) given by

$$\min_{x\in X} \Phi(x) := h(x) + r(x), \tag{1}$$

where $X \subseteq \mathbb{R}^d$ is convex and compact, $h : X \to \mathbb{R}$ is continuously differentiable but possibly non-convex, and r is a regularizer that potentially can be non-convex and non-differentiable, such as MCP [19], SCAD [18] or Capped ℓ_1 [31] etc. Here we assume that the gradient of h is L_h-Lipschitz continuous and only noisy (stochastic) gradients of h are available.

Problem (1) covers a wide range of non-convex or convex optimization problems. For examples, given a sequence of n examples $\{(a_i, b_i)\}_{i=1}^n$, where $a_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$, let $h(x) = \frac{1}{n}\sum_{i=1}^n [\varphi(a_i^T x - c) - b_i]^2$ and $r(\cdot) \equiv 0$, where c is a bias and $\varphi(\cdot)$ is a sigmoid function, i.e., $\varphi(t) = 1/(1+\exp(-t))$, then (1) reduces to the classic Perceptron which is non-convex; or let $h(x) = \frac{1}{n}\sum_{i=1}^n (a_i^T x - b_i)^2$, $r(\cdot) = \lambda\|\cdot\|_1$, $\lambda > 0$, then we obtain the well known Lasso [32] or Compressed Sensing [33]. However, as pointed out in [18], convex penalties lack the oracle property and lead to biased solutions. Compressed Sensing theory also shows that non-convex penalties can tolerate "bad" measure matrix $A := [a_1, \cdots, a_n]^T$ of which the restricted isometry properties (RIP) condition [34] might be slacked.

In this work we aim to develop efficient asynchronous stochastic optimization algorithms to solve Problem (1) where h is smooth and non-convex, and r is non-convex and non-differentiable ("non-convex+non-convex"). The proposed algorithms allow multiple processors to work independently with diverse working paces, where each processor only needs to call the stochastic first-order oracle ($\mathcal{SFO}$) or the stochastic zeroth-order oracle ($\mathcal{SZO}$) to get noise gradients.

1.1. **Related Work.** Stochastic optimization problems have been intensely studied and part of well known results of interest are summarized in Table 2. Particularly, when $h(\cdot)$ is both strongly convex and differentiable, and $r(\cdot) \equiv 0$, [35] showed that, by running SGD for at most $\mathcal{O}(1/\epsilon)$ iterations, one can find a point $\mathring{x}$ such that $\mathbb{E}[\Phi(\mathring{x})] - \Phi^* \leq \epsilon$, where $\Phi^* := \min_{x\in X}\Phi(x)$. When $h(\cdot)$ is non-convex but differentiable, [36] proposed a randomized stochastic gradient (RSG) method with which one can find an ϵ-solution, i.e., a point $\mathring{x}$ such that $\mathbb{E}[\|\nabla\Phi(\mathring{x})\|^2] \leq \epsilon$, with at most $\mathcal{O}(1/\epsilon^2)$ calls to $\mathcal{SFO}$s. RSG considers only the unconstrained scenario and there is no convergence guarantee for cases when $X \neq \mathbb{R}^d$ or $r(\cdot)$ is non-differentiable. To handle these problems, [37] further proposed a modified version of RSG, called the randomized stochastic proxima gradient (RSPG), where a mini-batch of samples are visited in each iteration. With at most $\mathcal{O}(1/\epsilon^2)$ calls to $\mathcal{SFO}$s, RSPG can find a solution $\mathring{x} \in X$ such that $\mathbb{E}[\|g_X(\mathring{x})\|^2] \leq \epsilon$, where $g_X(\mathring{x})$ is a generalized projected gradient of Φ at $\mathring{x}$.

Without considering the regularizer $r(\cdot)$, [15] proposed an asynchronous parallel stochastic gradient (ASYSG) algorithm and the analysis therein showed that by running

TABLE 1. Some recent methods in the stochastic and distributed optimization.

Reference	$h(\cdot)$	$r(\cdot)$	Parallel	Asynchronous	Convergence rate	Criterion of convergence
SGD [35]	strong convex	0	no	no	$\mathcal{O}(1/\epsilon)$	$\mathbb{E}[\Phi(\mathring{x})] - \Phi^* \leq \epsilon$
RSG [36]	nonconvex	0	no	no	$\mathcal{O}(1/\epsilon^2)$	$\mathbb{E}[\|\nabla\Phi(\mathring{x})\|^2] \leq \epsilon$
RSPG [37]	nonconvex	convex	no	no	$\mathcal{O}(1/\epsilon^2)$	$\mathbb{E}[\|g_X(\mathring{x})\|^2] \leq \epsilon$
ASYSG [15]	convex	0	yes	yes	$\mathcal{O}(1/\epsilon^2)$	$\mathbb{E}[\Phi(\mathring{x})] - \Phi^* \leq \epsilon$
ASYSPG [16]	convex	convex	yes	yes	$\mathcal{O}(1/\epsilon^2)$	$\mathbb{E}[\Phi(\mathring{x})] - \Phi^* \leq \epsilon$
ASYSPG [16]	strong convex	convex	yes	yes	$\mathcal{O}(1/\epsilon)$	$\mathbb{E}[\Phi(\mathring{x})] - \Phi^* \leq \epsilon$
ASYSG [11]	non-convex	0	yes	yes	$\mathcal{O}(1/\epsilon^2)$	$\mathbb{E}[\|\nabla\Phi(\mathring{x})\|^2] \leq \epsilon$
ARSPG (our work)	nonconvex	nonconvex	yes	yes	$\mathcal{O}(1/\epsilon^2)$	$\mathbb{E}[\|g_X(\mathring{x})\|^2] \leq \epsilon$
ARSPGF (our work)	nonconvex	nonconvex	yes	yes	$\mathcal{O}(1/\epsilon^2)$	$\mathbb{E}[\|g_X(\mathring{x})\|^2] \leq \epsilon$

ASYSG for at most $\mathcal{O}(1/\epsilon^2)$ iterations one can find a point $\mathring{x}$ such that $\mathbb{E}[\Phi(\mathring{x})] - \Phi^* \leq \epsilon$. [16] further extended this idea to a general convex composite problem where a convex regularizer $r(\cdot)$ is included, termed the asynchronous parallel stochastic proximal gradient (ASYSPG). When $h(\cdot)$ is strongly convex, [16] provided a sharper convergence rate, i.e., it can meet the convergence criterion with at most $\mathcal{O}(1/\epsilon)$ iterations. Moreover, [11] extended ASYSG to non-convex optimization without $r(\cdot)$, and the resulted algorithm can find an ϵ-solution with at most $\mathcal{O}(1/\epsilon^2)$ calls to $\mathcal{SFO}$s.

Notice that, all the methods mentioned above do not specially consider the general case where a non-convex regularization, such as MCP [19], SCAD [18] or Capped ℓ_1 [31] is used. Algorithms proposed in this paper are summarized as well in Table 2.

1.2. **Contributions.** In this paper, we first propose an algorithm with asynchronous randomized stochastic proximal gradient (ARSPG), which allows each processor to work with its own pace to solve general optimization problems of "non-convex+non-convex". Then, we specialize ARSPG to get a gradient-free algorithm (ARSPGF for short), where only the stochastic zeroth-order information is required. That is, ARSPGF is applicable to the case where the concrete form of $h(\cdot)$ is unavailable, as often occurs in machine learning problems [5, 38, 39, 40]. Specifically, the main contributions of the paper are summarized as follows.

- When $h(\cdot)$ is non-convex, most existing work [11, 41, 42] consider mainly the case where $r(\cdot)$ is convex or not included. The proposed method ARSPG is applicable to the general setting where a non-convex and non-smooth regularizer $r(\cdot)$ is used. The analysis shows that ARSPG can find a ϵ-solution with at most $\mathcal{O}(1/\epsilon^2)$ calls to $\mathcal{SFO}$s.
- In the case when only noisy function values of $h(\cdot)$ are available, e.g., $h(x) = \int \ell(x;\xi)\mathrm{d}\mathbb{P}(\xi)$, of which only limited accesses to $\{\ell(\cdot;\xi_i)\}_{i=1}^n$ is available, where $\{\xi_i\}_{i=1}^n$ are i.i.d. with an unknown distribution [43], the proposed method ARSPGF can achieve the same convergence rate (with a larger constant) as that of ARSPG, by using the stochastic zeroth-order information.
- In the case where both $h(\cdot)$ and $r(\cdot)$ are convex, ARSPG covers ASYSPG [16] which is one of the state of the art parallel methods for "convex + convex" SCO problems, and ARSPGF is applicable to the case where only $\mathcal{SZO}$ is available.

The remainder of the paper is organized as follows. Section 2 provides some notions and preliminaries. The method of ARSPG is proposed and analyzed in Section 3. The gradient-free version, ARSPGF is discussed in Section 4. Experiments carried on several benchmark data sets are presented in Section 5. The last section gives some concluding remarks.

2. **Notions and Preliminaries.** For simplicity, let $[N] := \{1, \cdots, N\}$ and $\xi_{[N]} := \{\xi_1, \cdots, \xi_N\}$. $\|\cdot\|$ denotes a general norm, and $\|\cdot\|_p$ is the standard p-norm for any $p \geq 1$, i.e., $\|x\|_p^p = \sum_{i=1}^d |x(i)|^p$, $\forall x \in \mathbb{R}^d$, where $x(i)$ denotes the ith element of x. Given any $X \subseteq \mathbb{R}^d$, we say $f \in \mathcal{C}_L(X)$, if f is Lipschitz continuously differentiable with Lipschitz constant $L > 0$, i.e., $\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|$, $\forall x, y \in X$. It is easy to verify that, when $f \in \mathcal{C}_L(X)$, for any $x, y \in X$,

$$|f(y) - f(x) - \langle \nabla f(x), y - x\rangle| \leq \frac{L}{2}\|y - x\|^2. \tag{2}$$

A function $f : X \to \mathbb{R} \cup \{+\infty\}$ is convex if it holds that for any $\lambda \in [0, 1]$, $\lambda f(x) + (1 - \lambda)f(y) \geq f(\lambda x + (1 - \lambda)y)$, $\forall x, y \in X$. A function $f : X \to \mathbb{R} \cup \{+\infty\}$ is β-strongly convex w.r.t. a norm $\|\cdot\|$, if for some $\beta \geq 0$, it holds that

$$f(y) - f(x) - \langle \nabla f(x), y - x\rangle \geq \frac{\beta}{2}\|y - x\|^2, \ \forall x, y \in X.$$

We also introduce several definitions for proper (possibly non-convex) functions.

Given a convex compact set $X \subseteq \mathbb{R}^d$, the following projection problem,

$$x^+ = \arg\min_{u \in X} \left\{ \langle \upsilon, u\rangle + \frac{1}{2\eta}\|u - x\|^2 + r(u) \right\}, \tag{3}$$

is easily solvable for any $\eta \geq \overline{\eta}$, $\upsilon \in \mathbb{R}^d$ and $x \in X$, where $\overline{\eta}$ is the largest value of η such that $\frac{1}{2\eta}\|\cdot\|^2 + r(\cdot)$ is convex. Let $\Phi(x) := h(x) + r(x)$, then the projected gradient of $\Phi(\cdot)$ at x can be written as

$$g_X(x, \upsilon, \eta) = \frac{1}{\eta}(x - x^+). \tag{4}$$

Actually, when $X \equiv \mathbb{R}^d$ and $r(\cdot) \equiv 0$, it reduces to the normal gradient, i.e., $g_X(x, \nabla h(x), \eta) = \nabla h(x) = \nabla \Phi(x)$.

In the rest part of the paper we make the following assumptions. First, we consider Problem (1) in the case where $h \in \mathcal{C}_{L_h}(X)$ (L_h is the Lipschitz constant of ∇h) and only the noisy gradient of h is available by calling sequentially to $\mathcal{SFO}$s. Specifically, at the k-th call, $\mathcal{SFO}$ will output a stochastic gradient $\nabla h(x_k; \xi_k)$ for the input $x_k \in X$, where ξ_k is a random variable that follows a certain distribution. For $\nabla h(x_k; \xi_k)$ we further make the following assumption.

Assumption 1. *For any $k \geq 1$,*

($\boldsymbol{A}1$) $\mathbb{E}[\nabla h(x_k; \xi_k)] = \nabla h(x_k)$;

($\boldsymbol{A}2$) $\mathbb{E}[\|\nabla h(x_k; \xi_k) - \nabla h(x_k)\|^2] \leq \sigma^2$.

3. **Asynchronous Stochastic First-order Method.** In this section, we present an asynchronous mini-bach algorithm that exploits multiple processors to solve the SCO problme (1) where both $h(\cdot)$ and $r(\cdot)$ are potentially non-convex. Theoretical aspects of the proposed algorithm are also discussed here.

3.1. **Synchronous Algorithm.** We first recall the synchronous protocol [37]. Assume that there are in all m processers in each iteration have access to the master server to exchange information. In particular, given $x_k \in X$, the i-th processer makes a call for $\mathcal{SFO}$ and get a feedback of stochastic gradient $\nabla h(x_k; \xi_{k,i})$. Generally synchronous parallelism is realized by repeating the following steps.

1. Given that the decision variable x_k is ready in the master server at the k-th iteration

.

2. The m processors make calls respectively for $\mathcal{SFO}$s, i.e., read x_k from the master and calculated the corresponding stochastic gradients $\nabla h(x_k;\xi_{k,i})$, $i \in [m]$.
3. When all the stochastic gradients are ready, the master calculates the averaged stochastic gradient, i.e., $H_k = \frac{1}{m}\sum_{i=1}^{m} \nabla h(x_k;\xi_{k,i})$, and updates the decision variable with

$$x_{k+1} = \arg\min_{x\in X}\left\{\langle H_k, x\rangle + \frac{1}{2\eta_k}\|x - x_k\|^2 + r(x)\right\}.$$

Notice that, in the above scheme, in order to update the decision variable, the master server has to wait until all the up-to-data stochastic gradients from m processors are available. However, in practice, the computation ability or the communication cost may vary with processors, and some of them may not respond to the master at the right time. Consequently, the optimizing process would be dominated by the weakest processor. Therefor we turn to asynchronous parallelism, with which the decision variable can be updated as long as at least one of the stochastic gradients is available.

3.2. **Asynchronous Algorithm.** In this subsection, we formally introduce the asynchronism following the spirit of [13, 14] and develop it to solve "non-convex+non-convex" problems.

In particular, let $m_k \leq m$ denote the total number of processors which are available for the master at the k-th iteration, and correspondingly let $\tau_{k,i}$ denote the time-varying delay for the i-th processor. Let us consider an example as illustrated in Figure 1, where $G_{i,k}$ denotes the stochastic gradient $\nabla h(x_k;\xi_{k,i})$ for short. From Figure 1, one can see that $\{m_1,\ldots,m_4\} = \{2,1,3,3\}$, $\{\tau_{1,1},\ldots,\tau_{4,1}\} = \{0,0,0,0\}$, $\{\tau_{1,2},\ldots,\tau_{3,2}\} = \{0,1,1\}$ and $\{\tau_{1,3},\tau_{2,3}\} = \{2,2\}$.

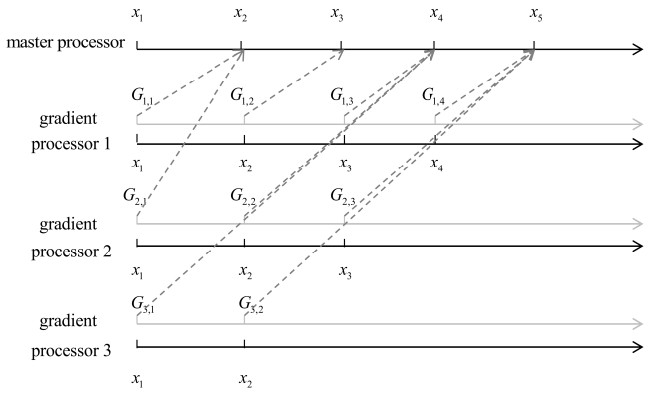


FIGURE 1. Illustration of a asynchronous protocol.

To explore theoretical aspects of the proposed algorithms, we further made the following assumption.

Assumption 2 (Bounded delay)**.** *For any $k \geq 1$ and $1 \leq i \leq m$, there exists a nonnegative integer τ such that*

$$(\boldsymbol{A}3)\ 0 \leq \tau_{k,i} \leq \tau.$$

Assumption (**A**3) implies that the master can receive the response from the processors at least once within the period $[k-\tau-1, k]$ where $k > \tau + 1$.

To develop the specific asynchronous algorithm, we modify the third step of the above synchronous version by replacing $\langle H_k, x\rangle$ with $\langle H_k, x\rangle + \triangle h_k$ where $\triangle h_k := h(x_k) -$

$\langle H_k, x_k\rangle$ and

$$H_k = \frac{1}{m_k}\sum_{i=1}^{m_k}\nabla h(x_{k-\tau_{k,i}};\xi_{k,i}). \tag{5}$$

That is, we update x_k with

$$x_{k+1} = \arg\min_{x\in X}\left\{\langle H_k, x\rangle + \triangle h_k + r(x) + \frac{1}{2\eta_k}\|x - x_k\|^2\right\}. \tag{6}$$

The resulted algorithm (called the asynchronous randomized stochastic projected gradient (ARSPG)) is formally presented in Algorithm 1. Notice that the term $\triangle h_k$ in (6) can be removed in optimization but it is helpful for theoretical analysis.

Algorithm 1 ARSPG algorithm.

1: **Input:** An initial point $x_1 \in X$, an iteration limit N, positive stepsizes $\{\eta_k\}_{k=1}^N$ and a random number $R \in [N]$;
2: **for** $k = 1, 2, \ldots, R$ **do**
3: **For the ith processor:**
4: Call x_t from the master server, calculate $\nabla h(x_k;\xi_{k,i})$ and send it back.
5: **For the master server:**
6: Receive m_k stochastic gradients from m_k processors and calculate H_k with (5);
7: Update x_{k+1} with (6);
8: **end for**

There are two noticeable differences in Algorithm 1, compared with the synchronous protocol. At first, instead of updating the decision variable N times, a randomized stopping criteria is used to avoid unnecessary computations. Actually, it is regular for Stochastic Programming, as shown in [44], to update first the decision variable N times and then output one of them randomly, e.g., to output x_k as the solution where k is uniformly sampled from $[N]$. In this case, $x_{k+1}, \cdots, x_N$ are useless and the computations caused by them are unnecessary. Secondly, with the asynchronism, ARSPG does not require that $m_k \equiv m$, and then the idle time can be reduced and more update can be carried out. Interestingly, as it will be shown in the next subsection, even in the case where there is only one processor is available at some iterations, Algorithm 1 can still guarantee a nearly optimal convergence rate when both $h(\cdot)$ and $r(\cdot)$ are convex.

3.3. **Theoretical Aspects of ARSPG.** Now we turn to the theoretical aspects of Algorithm 1. The main results are formally state as follows where the expectation is taken with respect to R and $\xi_{[N]}$.

Theorem 3.1. *Let $h \in \mathcal{C}_{L_h}(X)$ and Assumptions (**A**1)-(**A**3) are satisfied.*

(a) Case I: h and r in Problem (1) are non-convex and there exists a constant $\overline{\eta}$ such that $\frac{1}{2\overline{\eta}}\|\cdot\|^2 + r(\cdot)$ is β-strongly convex. Let $\eta_k < \min\left\{\frac{1}{\tau^2 L_h^2 + L_h + 1}, \overline{\eta}\right\}$, and for any $k \in [N]$,

$$Prob\{R = k\} = \frac{\eta_k - (\tau^2 L_h^2 + L_h + 1 - \beta)\eta_k^2}{\sum_{k=1}^N [\eta_k - (\tau^2 L_h^2 + L_h + 1 - \beta)\eta_k^2]}. \tag{7}$$

Then we have

$$\mathbb{E}\left[\|g_X(x_R, H_R, \eta_R)\|^2\right] \le \frac{\Phi(x_1) - \Phi^* + \frac{2\sigma^2}{\beta}\sum_{k=1}^N \frac{1}{m_k}}{\sum_{k=1}^N [\eta_k - (\tau^2 L_h^2 + L_h + 1 - \beta)\eta_k^2]}. \tag{8}$$

(b) Case II: h and r in Problem (1) are convex and x^* is one of its optimal solutions. Let $\eta_k < \frac{1}{L_h(\tau+1)^2}$, and for any $k \in [N]$

$$Prob\{R = k\} = \frac{1}{N}. \tag{9}$$

Then we have

$$\mathbb{E}[\Phi(x_R)] - \Phi^* \leq \frac{\sum_{k=1}^N \frac{\sigma^2}{2[1/\eta_k - L_h(\tau+1)^2]m_k} + \frac{1}{2\eta_1}\|x^* - x_1\|^2}{N}.$$

The proof of Theorem 3.1 is presented in Appendix 7.1.

Remark 3.1. *For case II, Theorem 3.1 provides a nearly optimal convergence rate, i.e., $\mathcal{O}(1/\epsilon^2)$, even in the case where $m_k = 1$ for some $k \in [N]$. Actually, let $\eta_k \equiv \frac{c_1}{\sqrt{N}}$, $0 < c_1 < \frac{1}{L_h(\tau+1)^2}$, it is easy to verified that $\mathbb{E}[\Phi(x_R)] - \Phi^* \leq \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$.*

Remark 3.2. *In case I, there is no any convexity limitation for the regularizer and Theorem 3.1 is applicable to the general case where a non-convex sparsity regularizer such as MCP, SCAD or Capped ℓ_1 cis used. Let us take MCP for example, i.e., $r(x) = \sum_{i=1}^d \phi_{\lambda,\kappa}(x(i))$ and*

$$\phi_{\lambda,\kappa}(t) = \begin{cases} \lambda\left(|t| - \dfrac{t^2}{2\lambda\kappa}\right) & |t| < \kappa\lambda, \\ \dfrac{\lambda^2\kappa}{2} & |t| \geq \kappa\lambda, \end{cases} \tag{10}$$

where $\lambda, \kappa > 0$. From (10), we know that there exists a constant $\overline{\eta} = \frac{\kappa}{1+2\beta\kappa}$ such that $\frac{1}{2\overline{\eta}}\|\cdot\|^2 + r(\cdot)$ is β-strongly convex. To run Algorithm 1, we can let $\eta_k \equiv c_2 < \min\left\{\frac{1}{\tau^2 L_h^2 + L_h + 1}, \frac{\kappa}{1+2\beta\kappa}\right\}$ and $Prob\{R = k\} = \frac{1}{N}$.

Further, let $0 < c_3 < \min\left\{\frac{1}{\tau^2 L_h^2 + L_h + 1}, \overline{\eta}\right\}$ and replace η_k in (8) with c_3, we have the following result.

Corollary 3.1. *Let $h \in \mathcal{C}_{L_h}(X)$ and Assumptions (**A**1)-(**A**3) are satisfied. Consider Case I as stated in Theory 3.1. Let $\eta_k \equiv c_3, \forall k \in [N]$, and R is generated following (7). Then we have*

$$\mathbb{E}\left[\|g_X(x_R, H_R, \eta_R)\|^2\right] \leq \frac{1}{\mathcal{B}_1}\left[\frac{\Phi(x_1) - \Phi^*}{N} + \frac{2\sigma^2}{\beta N} \cdot \sum_{k=1}^N \frac{1}{m_k}\right],$$

where $\mathcal{B}_1 = c_3\left[1 - (\tau^2 L_h^2 + L_h + 1 - \beta)c_3\right]$.

Remark 3.3. *From Corollary 3.1, we can see that the right side of the above result will be dominated by the second term when N is sufficiently large. Let $\sum_{k=1}^N \frac{1}{m_k} = o(N)$, then one can expect that $\mathbb{E}\left[\|g_X(x_R, H_R, \eta_R)\|^2\right] \to 0$. In particular, if we make an assumption as also used in [37] that $\sum_{k=1}^N \frac{1}{m_k} \leq \mathcal{O}(\sqrt{N})$, then the total number of calls to $\mathcal{SFO}$s is $\mathcal{O}(1/\epsilon^2)$ to find a solution $x_R \in X$ such that $\mathbb{E}\left[\|g_X(x_R, H_R, \eta_R)\|^2\right] \leq \epsilon$.*

4. **Asynchronous Stochastic Zeroth-order Method.** In applications, there exist cases where only noisy function values of $h(\cdot)$ are available, for which gradient-based methods are not directly applicable. Let us consider an example of regression where a direct calculation of the likelihood is infeasible. Specifically, let $h(\cdot)$ be the negative log-likelihood of a conditional Gibbs measure π_x. In this case the gradient of $h(\cdot)$ is typically expressed as $\nabla h(x) = \int \ell(x;\xi)\mathrm{d}\pi_x(\xi)$, where ξ denotes the latent variable and $\ell(\cdot;\cdot)$ the

loss function for the regression. Since the exactly Gibbs measure π_x is not available, the integral cannot be calculated in closed form [45].

Formally, we assume that the processors only have access to the the noisy zeroth-order information of $h(\cdot)$ by calling the stochastic zeroth-order oracle ($\mathcal{SZO}$). That is, at the k-th iteration, only a stochastic function value $h(x_k;\xi_k)$ is available, where ξ_k is a random variable whose distribution is supported on a set $\Xi_k \subseteq \mathbb{R}^d$. Throughout this section, we make the following assumption for the unbiasedness of $\mathcal{SZO}$.

Assumption 3. *For any $k \geq 1$,*

$$(\boldsymbol{A}4)\ \mathbb{E}[h(x_k;\xi_k)] = h(x_k).$$

To develop a feasible algorithm, we first introduce several basic concepts. A smooth approximation of $h(\cdot)$ is defined as

$$h_\mu(x) = \int h(x+\mu\zeta)\rho(\zeta)\mathrm{d}\zeta,$$

where $\mu > 0$ is the smoothing parameter, $\zeta \in \mathbb{R}^d$ is a random vector with the density function $\rho(\cdot)$. When $\rho(\cdot)$ is Gaussian, [46] showed that the resulted Gaussian smoothing approximation has interesting properties (see Lemma 7.1 for details) which support the theoretical analysis here. Inspired by the smooth approximation of $h(\cdot)$, we can further introduce an approximation for $\nabla h(x_k)$, i.e.,

$$\nabla h_\mu(x_k;\xi_k,\zeta) := \frac{h(x_k+\mu\zeta;\xi_k) - h(x_k,\xi_k)}{\mu}\zeta, \tag{11}$$

where ζ is sampled from Gaussian.

With the approximation (11), we can extend Algorithm 1 to the case where the exact form of $h(\cdot)$ is not revealed but the function values of $h(\cdot)$ are acquirable. The resulted algorithm (called the asynchronous randomized stochastic proximal gradient free (ARSPGF) algorithm) is presented in Algorithm 2

Algorithm 2 ARSPGF Algorithm.

1: **Input:** An initial point $x_1 \in X$, an iteration limit N, positive stepsizes $\{\eta_k\}_{k=1}^N$ and a random number $R \in [N]$;
2: **for** $k = 1, 2, \ldots, R$ **do**
3: **For the ith processor:**
4: Call x_t from the master server, calculate $\nabla h_\mu(x_k;\xi_{k,i},\zeta_{k,i})$ and send it back.
5: **For the master server:**
6: Receive m_k approximate stochastic gradients from m_k processors and calculate $H_{\mu,k}$ with

$$H_{\mu,k} = \frac{1}{m_k}\sum_{i=1}^{m_k} \nabla h_\mu(x_{k-\tau_{k,i}};\xi_{k,i},\zeta_{k,i});$$

7: Update x_{k+1} with

$$x_{k+1} = \arg\min_{x\in X}\left\{\langle H_{\mu,k}, x\rangle + \frac{1}{2\eta_k}\|x-x_k\|^2 + r(x)\right\};$$

8: **end for**

For the approximate stochastic gradient, by the result (a) stated in Lemma 7.1, it holds that

$$\mathbb{E}_{\zeta,\xi}[\nabla h_\mu(x;\xi,\zeta)] = \nabla h_\mu(x). \tag{12}$$

Moreover, by the result (b) stated in Lemma 7.1, $\nabla h_\mu(x)$ is a good approximation of $\nabla h(x)$ when μ is small enough. Hence, we can expect a similar performance as that of Algorithm 1. Formally, we have the following theoretical results, where the expectation is taken w.r.t. R, $\xi_{[N]}$ and $\zeta_{[N]}$

Theorem 4.1. *Let $h \in \mathcal{C}_{L_h}(X)$ and Assumptions (**A**1)-(**A**4) are satisfied, and for any $x \in X$, $\|\nabla h(x)\| \leq C$ where $C > 0$.*

(a) Case I: h and r in Problem (1) are non-convex. η_k and R are the same as that stated in Theorem 3.1 (a). Then $\mathbb{E}\left[\|g_X(x_R, H_{\mu,R}, \eta_R)\|^2\right]$ is upper bounded by

$$\frac{\Phi_\mu(x_1) - \Phi_\mu^* + \frac{2\widetilde{\sigma}^2}{\beta}\sum_{k=1}^N \frac{1}{m_k}}{\sum_{k=1}^N [\eta_k - (\tau^2 L_h^2 + L_h + 1 - \beta)\eta_k^2]},$$

where

$$i)\ \Phi_\mu(x) = h_\mu(x) + r(x),$$

$$ii)\ \widetilde{\sigma}^2 = 2(d+4)(C^2+\sigma^2) + \frac{\mu^2}{2}L_h^2(d+6)^2.$$

(b) Case II: h and r in Problem (1) are convex and x^* is one of its optimal solutions. η_k and R are the same as that stated in Theorem 3.1 (b). Then $\mathbb{E}[\Phi(x_R) - \Phi^*]$ is upper bounded by

$$\frac{\sum_{k=1}^N \frac{\widetilde{\sigma}^2}{2[1/\eta_k - L_h(\tau+1)^2]m_k} + \frac{1}{2\eta_1}\|x^* - x_1\|^2}{N} + \mu^2 d L_h,$$

where $\widetilde{\sigma}$ is the same as that defined in (a).

The proof can be found in Appendix 7.2. By Theorem 4.1, we further have the following results.

Corollary 4.1. *Let $h \in \mathcal{C}_{L_h}(X)$ and Assumptions (**A**1)-(**A**4) are satisfied, and for any $x \in X$, $\|\nabla h(x)\| \leq C$ where $C > 0$.*

(a) Consider Case I as stated in Theorem 4.1. Let $0 < c_3 < \min\left\{\frac{1}{\tau^2 L_h^2 + L_h + 1}, \overline{\eta}\right\}$, $\eta_k \equiv c_3$, $\forall k \in [N]$, and R is generated following (7). Then $\mathbb{E}\left[\|g_X(x_R, H_{\mu,R}, \eta_R)\|^2\right]$ is upper bounded by

$$\frac{1}{\mathcal{B}_1}\left[\frac{\Phi_\mu(x_1) - \Phi_\mu^*}{N} + \frac{2\widetilde{\sigma}^2}{\beta N}\cdot\sum_{k=1}^N \frac{1}{m_k}\right],$$

where $\mathcal{B}_1 = c_3\left[1 - (\tau^2 L_h^2 + L_h + 1 - \beta)c_3\right]$.

(b) Consider Case II as stated in Theorem 4.1. Let $c_1 < \frac{1}{L_h(\tau+1)^2}$, $\eta_k \equiv \frac{c_1}{\sqrt{N}}$, $\forall k \in [N]$, and R is generated following (9). Then $\mathbb{E}[\Phi(x_R) - \Phi(x^*)]$ is upper bounded by

$$\frac{1}{\sqrt{N}}\left[\frac{c_1\widetilde{\sigma}^2}{2[1 - c_1 L_h(\tau+1)^2]} + \frac{\|x^* - x_1\|^2}{2c_1}\right] + \mu^2 d L_h.$$

The proof for Corollary 4.1 is straightforward and we omit it here.

Remark 4.1. *If the assumption is hold that $\sum_{k=1}^N \frac{1}{m_k} \leq \mathcal{O}(\sqrt{N})$ and $\mu \leq \mathcal{O}\left(\frac{1}{\sqrt[4]{N}}\right)$, then the total number of calls to $\mathcal{SZO}$s is $\mathcal{O}(1/\epsilon^2)$ to find a solution $x_R \in X$ such that $\mathbb{E}\left[\|g_X(x_R, H_R, \eta_R)\|^2\right] \leq \epsilon$ for Case I or such that $\mathbb{E}[\Phi(x_R) - \Phi(x^*)] \leq \varepsilon$ for Case II. It is encouraging that, with the same order of convergence rate as that of Algorithm 1, Algorithm 2 can get a solution with the same accuracy .*

Remark 4.2. *What we have to pay for the absence of the exact form of $h(\cdot)$? Take a close look at the results presented above, one can notice that the above bounds included the dimension d. In particular, the variance of the approximate stochastic gradient is larger*

than that of the norma version, i.e., $\widetilde{\sigma}^2 > \sigma^2$. What is more, for Case II, there is an extra term $\mu^2 dL_h$ in the bound. It is not surprise that, with a larger constant for the convergence, $\mathcal{SZO}$-based methods may be slower than $\mathcal{SFO}$-based methods, and the large variance may cause extra fluctuations when d is very large (Interestingly, one will find that in the experiments its performance is not so sensitive to d).

Remark 4.3. *In practice, we may have some priori information about $h(\cdot)$ although we do not know its exact form. Therefor, it will be interesting to introduce some structures or constraints to reduce the variance for $\mathcal{SZO}$-based methods.*

5. **Experiments.** In this section, we test the applicability of the two propose algorithms ARSPG and ARSPGF, to solve"non-convex+non-convex" problems. Specifically, we test the algorithms of interest to solve the Perceptron where the sparse solutions are expected, by using the sparse promoting regularizer MCP. Given a sequences of examples $\{(a_j, b_j) \in \mathbb{R}^d \times \mathbb{R}\}_{j=1}^n$, we consider the following regularized objective functions:

$$\Phi(x) := \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+\exp(-a_i^T x + c)} - b_i\right)^2}_{h(x)} + \underbrace{\sum_{j=1}^{d}\phi_{\lambda,\kappa}(x(j))}_{r(x)},$$

where $\phi_{\lambda,\kappa}$ is the MCP penalty function as formulated in (10). The experiments are carried out on six benchmark data sets which are downloaded from LIBSVM[1] and UCI data repository [2] (see Table 2 [3] for details where `url` uses the first 10000 examples). We set $\kappa = 20$ for MCP in all the experiments. As for the regular parameter λ, which realizes the tradeoff between the sparsity of the solution and the ability to fit the data, a range of values are considered (see Table 3 for details).

TABLE 2. Data sets used in the experiments.

Data set	#Samples	#Features	#ANZF
url	10000	3231961	118
news20	19996	1355191	455
rcvl-2	20242	47236	74
real-sim	72309	20958	51
gisette	6000	5000	4955
mnist	60000	780	150

In the experiments, all algorithms are initialized with $x_0 = 0$, and $X = \mathbb{R}^d$ as also adopted in [37] and other references such as [11] and [41]. We set $m = 10$, $\tau = 50$, and to simulate the parallel scenario, m_k and $\tau_{k,i}$ are randomly sampled from $[m]$ and $[\tau]$ respectively. As for the stepsize, we need to estimate the Lipschitz constant L_h of ∇h. Specifically, we first sample a subset (denoted by Ω) of examples with equal probability, e.g, 100 examples from the data set. Then we calculate out the ℓ_2-norm of the Hessian of

[1] `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[2] `http://archive.ics.uci.edu/ml/datasets.html`

[3] the averaged non-zero features is shorted as ANZF in this table

$\Phi(\cdot)$ over the sampled examples, i.e.,

$$\frac{1}{|\Omega|}\sum_{i\in\Omega}\left[\frac{4\exp(-2(a_i^Tx-b_i))-2\exp(-(a_i^Tx-b_i))}{(1+\exp(-(a_i^Tx-b_i)))^4}\right.$$
$$\left.-2b_i\frac{\exp(-2(a_i^Tx-b_i))-\exp(-(a_i^Tx-b_i))}{(1+\exp(-(a_i^Tx-b_i)))^3}\right]a_ia_i^T.$$

We repeat the above process (sampling and calculating) five times and then take the average as the estimator of L_h. The resulted estimations of L_h are list in Table 3, where the smoothing parameter μ as used in zero-order algorithms is presented as well. The total iteration number $N = 2000$, and all the algorithms of interest will be run 20 times and the averaged performance will be reported.

Table 3. Parameters' values used in the experiments.

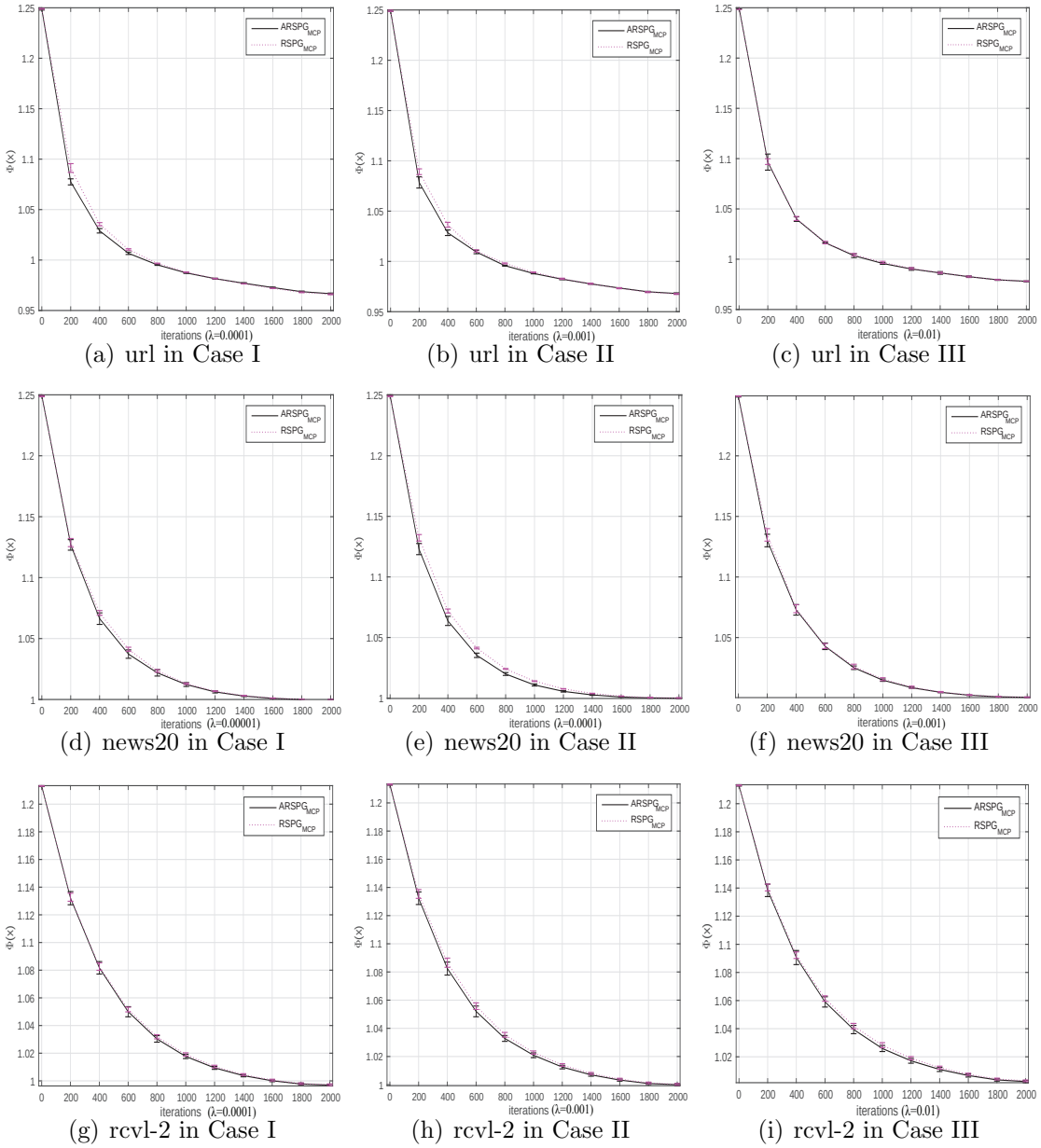
		λ			μ
Data set	L_h	Case I	Case II	Case III	
url	1.2256e+0	10^{-4}	10^{-3}	10^{-2}	$\frac{1}{\sqrt[4]{N}}e-01$
news20	1.8190e-01	10^{-5}	10^{-4}	10^{-3}	$\frac{1}{\sqrt[4]{N}}e-01$
rcvl-2	2.1280e-01	10^{-4}	10^{-3}	10^{-2}	$\frac{1}{\sqrt[4]{N}}e-01$
real-sim	1.9410e-01	10^{-5}	10^{-4}	10^{-3}	$\frac{1}{\sqrt[4]{N}}e-01$
gisette	7.6365e+0	10^{-3}	10^{-2}	10^{-1}	$\frac{1}{\sqrt[4]{N}}e-04$
mnist	5.2240e+02	10^{-2}	10^{-1}	10^{0}	$\frac{1}{\sqrt[4]{N}}e-03$

5.1. **Asynchronism vs. Synchronism.** First, we examine the performances of the algorithms motived respectively by the asynchronism and synchronism. In particular, the proposed algorithm ARSPG and the synchronous baseline RSPG [37] are tested. Notice that, RSPG is originally not a parallel algorithm and here we convert it into a parallel version.

Figures 2-3 show the behaviors of two algorithms over all the data sets involved. From the figures, we can see that ARSPG performs as well as (or even slightly better than) RSPG does. Notice that, for RSPG all the m processors are required to be available in each iteration, i.e., $m_k \equiv m = 10$. The results show that, with the asynchronism, the proposed algorithms ARSPG works well in a more general working condition, and naturally introduces slight perturbations (by using out-of-date gradients) which potentially improve its performance in practice. The results here validate well the analysis presented in [47] where the author argued that rather than using explicitly the immediate gradients, slightly perturbed gradients can improve the performances.

5.2. **First-order Method vs. Zeroth-order Method.** In practice there exist cases where only noisy function values of $h(\cdot)$ are available, for which we proposed a zeroth-order method, i.e., ARSPGF. Now we study its performance in practice. In particular, we examine the difference between the first-order algorithm (ARSPG) where noise gradients are available and the zeroth-order algorithm (ARSPGF) where only noise function values are available.

Figures 4-5 present the performances of two algorithms over all the data sets involved. From the figures we can observe that, without using exact gradients, ARSPGF performs as well as ARSPG does in most cases, and even better over data sets `news20`, `rcvl-2` and `gisette`. In particular, from Figure 5(f), one can find a huge gap between two



(a) url in Case I (b) url in Case II (c) url in Case III

(d) news20 in Case I (e) news20 in Case II (f) news20 in Case III

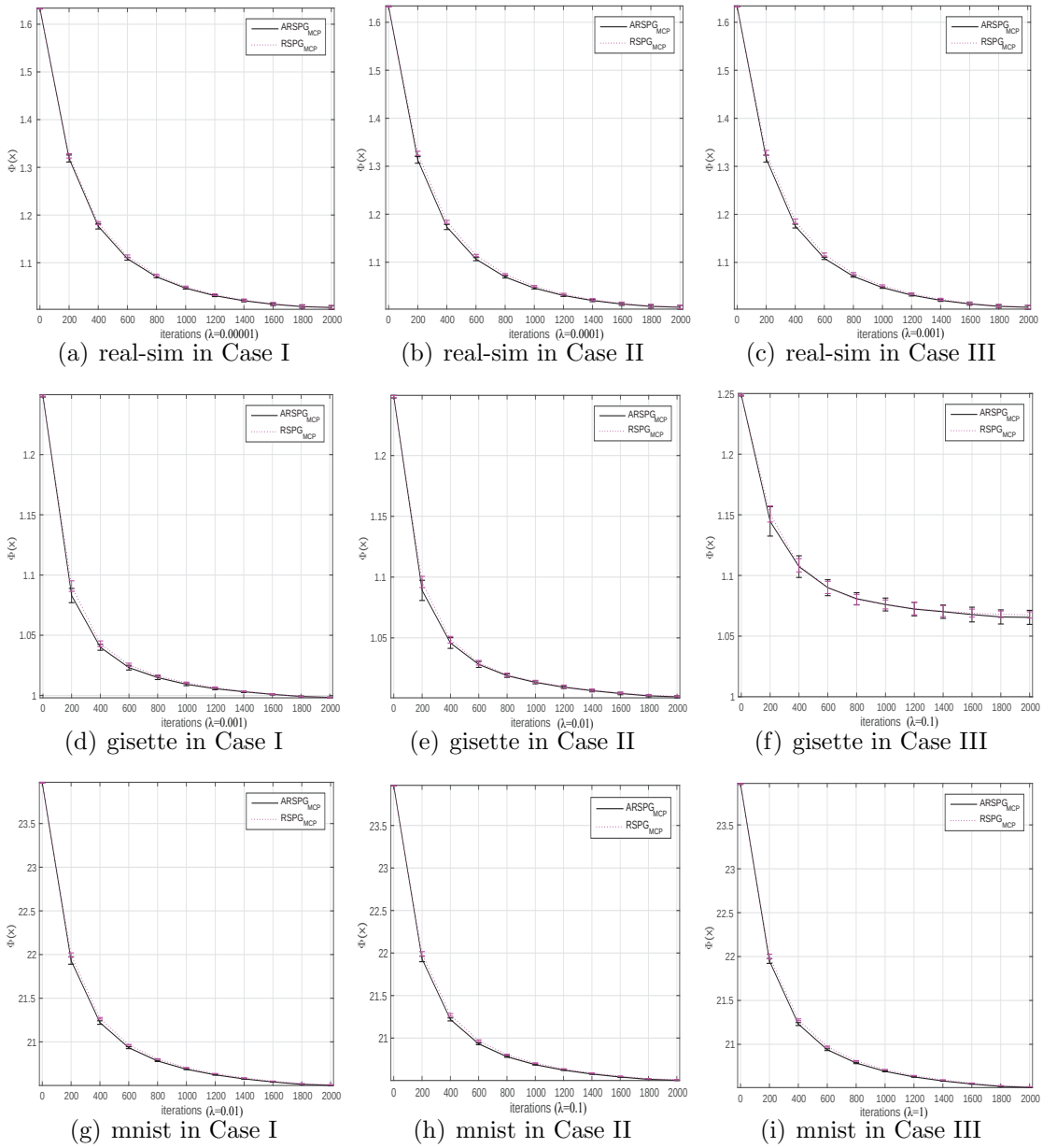
(g) rcvl-2 in Case I (h) rcvl-2 in Case II (i) rcvl-2 in Case III

FIGURE 2. Asynchronous algorithm ARSPG vs. synchronous algorithm RSPG over data sets `url`, `news20` and `rcvl-2`.

competitors, which shows that ARSPGF might be more robust to the values of regularizer parameter.

5.3. **Non-convex Regularizer vs. Convex Regularizer.** As mentioned earlier, most existing parallel algorithms mainly consider convex regularization for its theoretical tractability. However, in the view of statistics, convex regularization lacks the oracle property and leads to biased solutions. In this paper we introduce non-convex regularization, and to verify its theoretical results and examine its practical performance, here we compare specifically the non-convex regularizer MCP with the convex regularizer ℓ_1-norm.

Figures 6-7 show the performances of two methods over all the data sets of interest. Notice that in the figures $h(x)$, rather than $\Phi(x)$, is used to show the ability to capture the original objective to be optimized. Form the results we can see that, over three data sets,



(a) real-sim in Case I (b) real-sim in Case II (c) real-sim in Case III

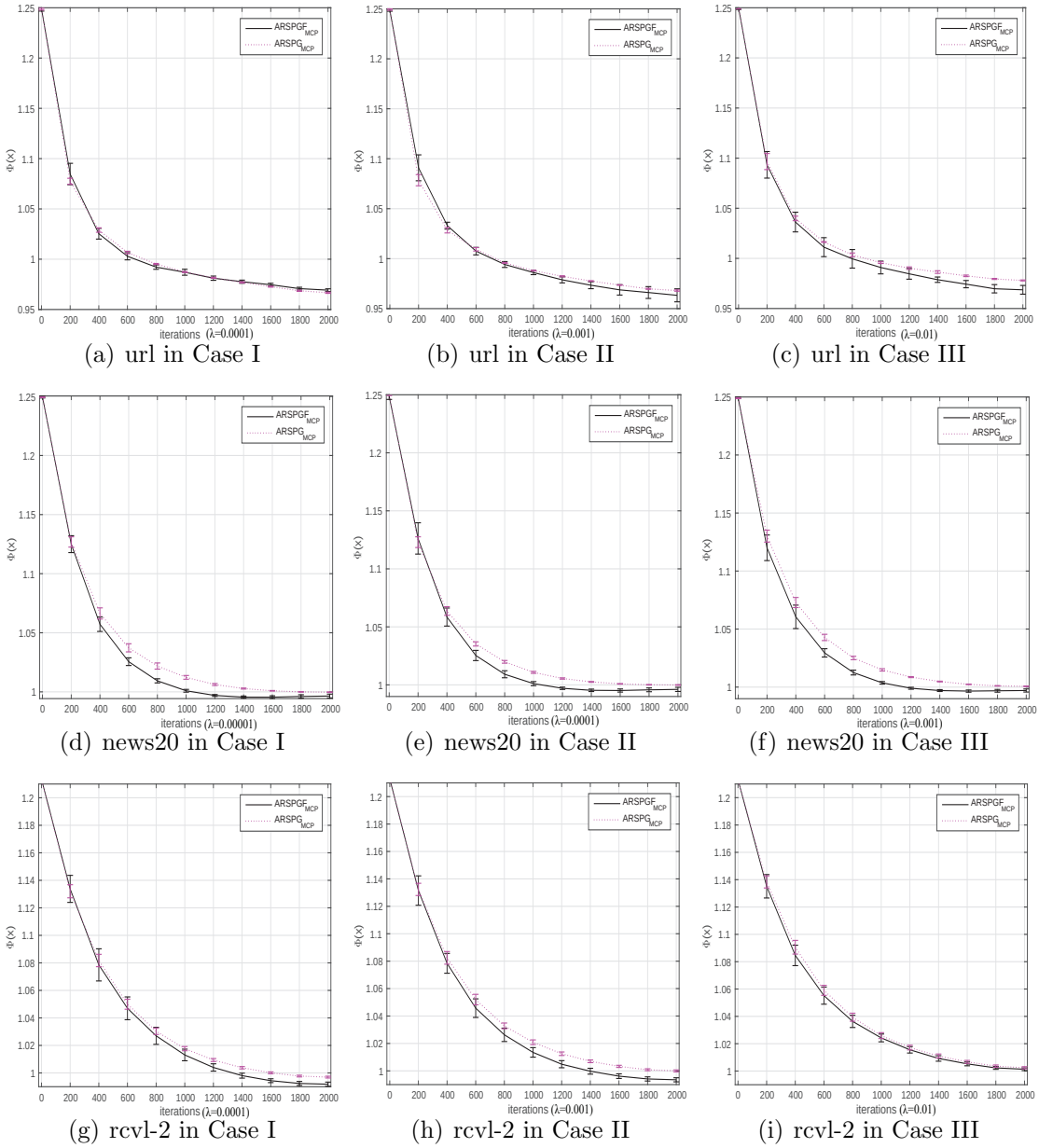
(d) gisette in Case I (e) gisette in Case II (f) gisette in Case III

(g) mnist in Case I (h) mnist in Case II (i) mnist in Case III

FIGURE 3. Asynchronous algorithm ARSPG vs. synchronous algorithm RSPG over data sets `real-sim`, `gisette` and `mnist`.

i.e., `news20`, `rcv1-2` and `real-sim`, MCP achieves lower values of $h(x)$ than ℓ_1-norm dose. The behaviors of the two algorithms over other data sets are similar. For the sparsity of solutions, Table 4 shows the details (the numbers of non-zero elements of solutions corresponds to different values of λ), from which, one can clearly see that, MCP tends to giving sparser solutions, compared with ℓ_1-norm. Notice that, MCP obtained the sparsity with lower (over 3 out of 6 data sets) or similar (over other 3 data sets) values of the objective function.

6. **Conclusion.** In this paper, we consider "non-convex + non-convex" SCO problems with the asynchronism. Based on the randomized stochastic proximal gradient method, we propose an asynchronous parallel algorithm termed ARSPG, which incorporates non-convex regularization well to obtain the oracle property. Theoretical analysis shows that



(a) url in Case I (b) url in Case II (c) url in Case III

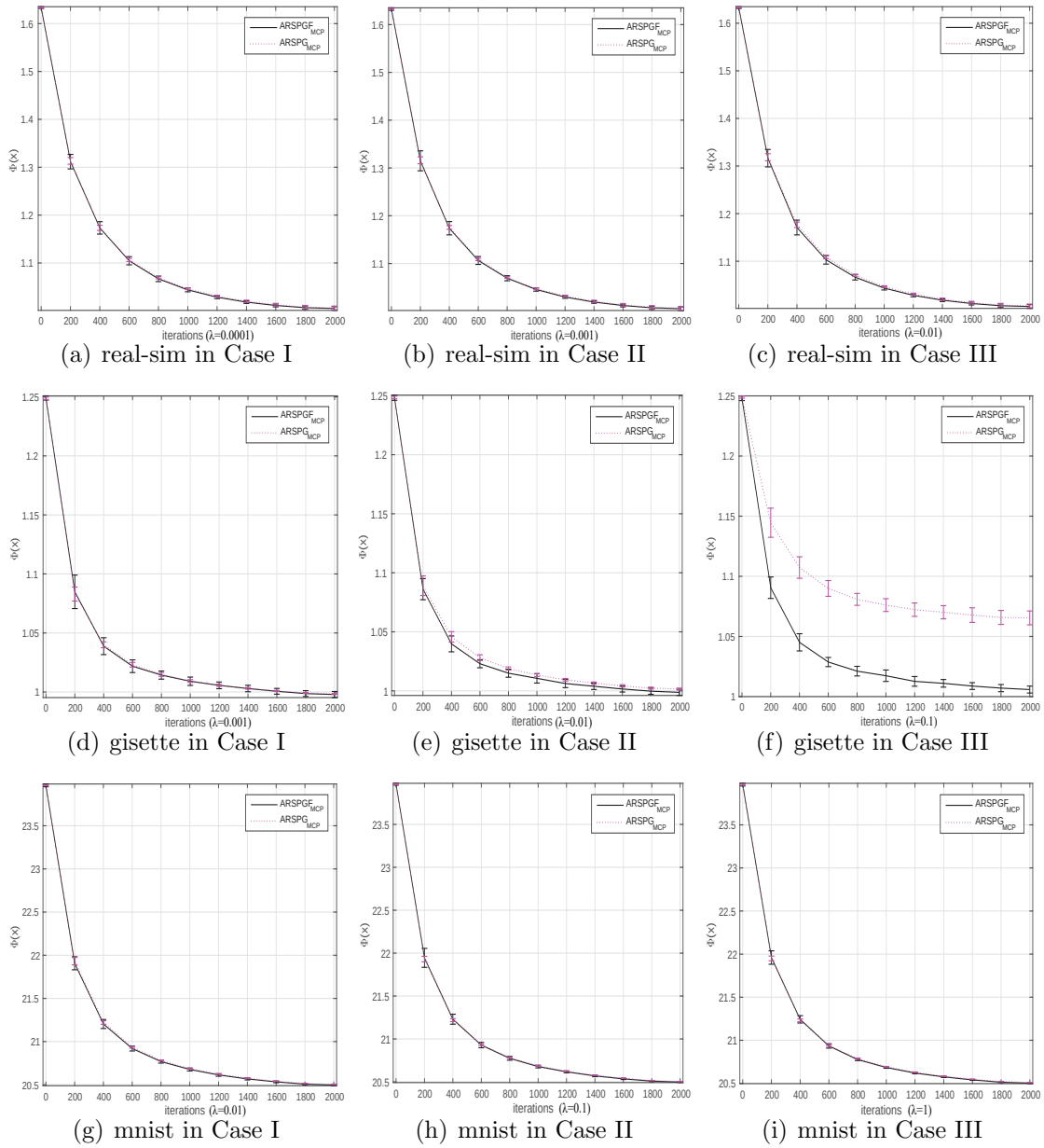
(d) news20 in Case I (e) news20 in Case II (f) news20 in Case III

(g) rcvl-2 in Case I (h) rcvl-2 in Case II (i) rcvl-2 in Case III

FIGURE 4. First-order algorithm ARSPG vs. zeroth-order algorithm ARSPGF over data sets `url`, `news20` and `rcvl-2`.

TABLE 4. Solutions' sparsity obtained by MCP and ℓ_1-norm regularizers.

	Case I		Case II		Case III	
Data set	MCP	ℓ_1-norm	MCP	ℓ_1-norm	MCP	ℓ_1-norm
url	19002±128.6	19343±158.0	3532±141.8	3476±208.3	621±50.7	622±67.1
news20	131030±6764.8	249360±14325.0	8177±2570.7	30012±3067.8	112±24.5	2802±350.3
rcvl-2	6114±193.7	7907±237.2	789±99.7	1471±134.7	10±7.7	51±21.5
real-sim	14135±117.1	16253±121.5	2421±47.1	4461±143.8	71±5.3	498±101.3
gisette	4955±0.0	4955±0.3	4953±1.5	4952±1.8	4522±162.2	4517±218.3
mnist	653±7.7	653±4.7	597±4.2	597±4.2	497±7.3	497±5.6



(a) real-sim in Case I (b) real-sim in Case II (c) real-sim in Case III

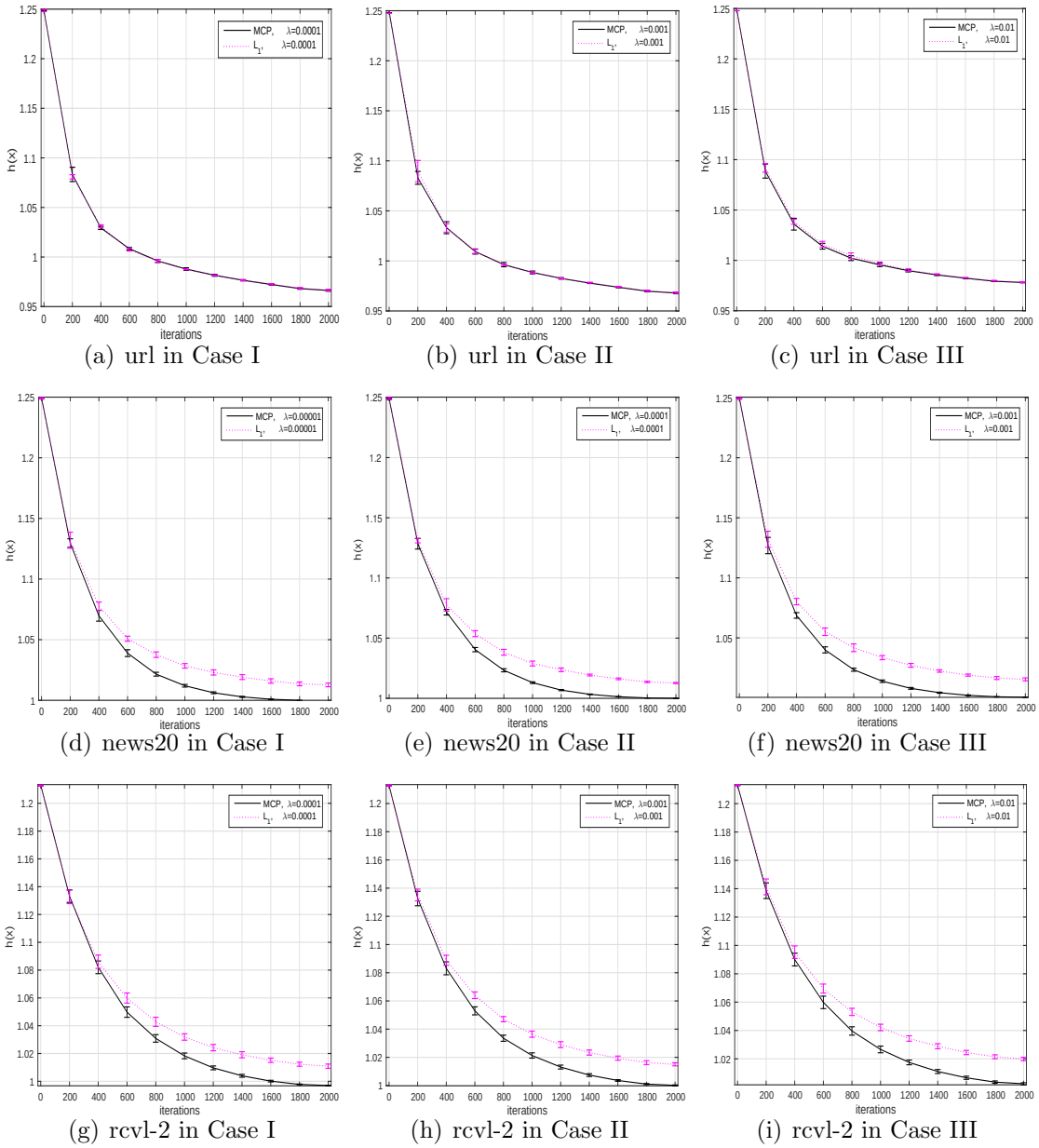
(d) gisette in Case I (e) gisette in Case II (f) gisette in Case III

(g) mnist in Case I (h) mnist in Case II (i) mnist in Case III

FIGURE 5. First-order algorithm ARSPG vs. zeroth-order algorithm ARSPGF over data sets `real-sim`, `gisette` and `mnist`.

ARSPG has the convergence rate in the order of $\mathcal{O}(1/\epsilon^2)$. Further, motivated by the cases where only noisy values of objective function are available and then gradient-based methods are not directly applicable, we extend the proposed algorithm to a gradient-free version, i.e., ARSPGF. Theoretical analysis shows that ARSPGF can achieve the same order of convergence rate (with a larger constant) as that of the normal version.

Methods proposed in this paper are examined from practical aspects as well. Experimental results over six benchmark data sets validate well the theoretical analyses. Interestingly we can observe that, slight perturbations introduced by the asynchronism make ARSPG perform even slightly better than its synchronous counterpart in some cases. Without using the exact gradients, ARSPGF performs no worse than the algorithm using the exact ones, but theoretically it may lead to extra fluctuations in cases when the dimensionality of d is very large. For non-convex regularization, experimental



(a) url in Case I (b) url in Case II (c) url in Case III

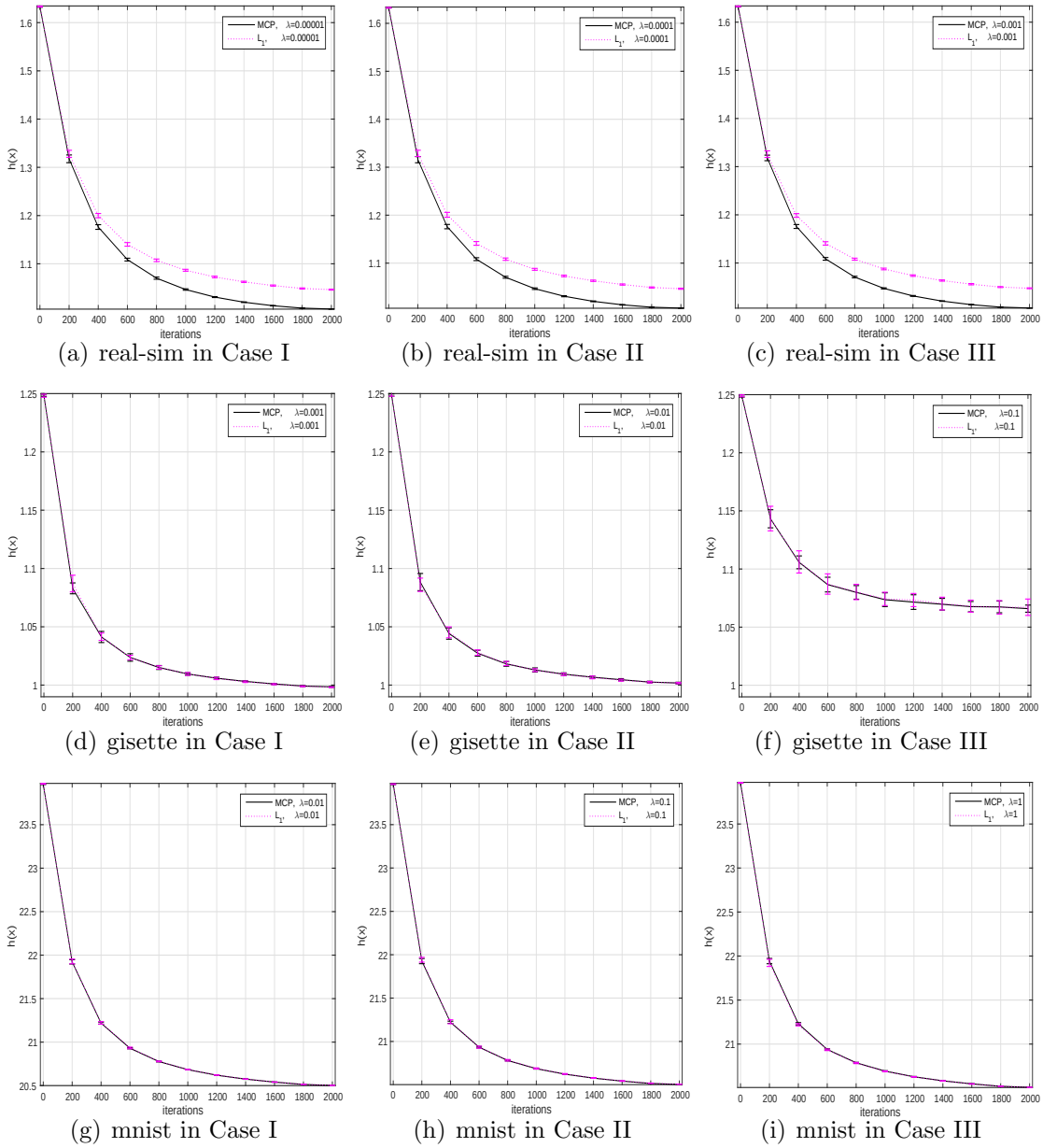
(d) news20 in Case I (e) news20 in Case II (f) news20 in Case III

(g) rcvl-2 in Case I (h) rcvl-2 in Case II (i) rcvl-2 in Case III

FIGURE 6. ARSPG+MCP vs. ARSPG+ℓ_1-norm over datasets `url`, `news20` and `rcvl-2`.

results are positive as well and the non-convex regularizer MCP outperforms the convex regularizer ℓ_1-norm, both in the sense of the ability to capture the true objective to be optimized and the sparsity of the solutions.

To solve SCO problems in parallel is strongly motivated in the age of big data. The work presented in this paper makes a try to solve "non-convex+non-convex" problems. There must have been much work to complete the study on this kind of topics. In the near future the following interesting issues will be explored: 1) decentralized optimization where no master server is necessary and the processors minimize collaboratively the sum of their local objective functions over the underlying topologies; 2) online version to handle the case where the objective function is time-varying; 3) $\mathcal{SZO}$-based method with reduced variance where the priori information, the structures or constraints of the objective function can be exploited; 4) further study on zero-order algorithms or the



(a) real-sim in Case I (b) real-sim in Case II (c) real-sim in Case III

(d) gisette in Case I (e) gisette in Case II (f) gisette in Case III

(g) mnist in Case I (h) mnist in Case II (i) mnist in Case III

FIGURE 7. ARSPG+MCP vs. ARSPG+ℓ_1-norm over datasets `real-sim`, `gisette` and `mnist`.

combination of exact gradient with objective function values calculation for better performance of composite optimization; 5) gradient compression to improve the communication efficiency of distributed optimization.

Acknowledgments. This work is supported by the Natural Science Foundation of Fujian Province under Grant 2020J01891 and Grant 2018H4005, and partly by the National Natural Science Foundation of China under Grant 41971340.

REFERENCES

[1] L. Bottou, "Stochastic gradient descent tricks," *Neural Networks: Tricks of the Trade, Reloaded*, pp. 421–436, 2012.

[2] W. He, J. T.-Y. Kwok, J. Zhu, and Y. Liu, "A note on the unification of adaptive online learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 5, pp. 1178–1191, 2017.
[3] J. Duchi, "Introductory lectures on stochastic optimization," *IAS/Park City Mathematics Series*, pp. 99–185, 2018.
[4] W. He and Y. Liu, "To regularize or not: Revisiting sgd with simple algorithms and experimental studies," *Expert Systems With Applications*, vol. 112, pp. 1–14, 2018.
[5] G. Lan, *First-order and stochastic optimization methods for machine learning.* Springer Nature Switzerland AG, Cham, 2020.
[6] Z. Lin, H. Li, and C. Fang, *Accelerated optimization for machine learning.* Springer Nature Singapore Pte Ltd, Singapore, 2020.
[7] A. Beck, *First-order methods in optimization.* Society for Industrial and Applied Mathematics, Philadelphia, 2017.
[8] W. Wang and N. Srebro, "Stochastic nonconvex optimization with large minibatches," *arXiv:1709.08728*, 2017.
[9] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev Soc Ind Appl Math*, vol. 60, no. 2, pp. 223–311, 2018.
[10] R. Zhang and J. T. Kwok, "Asynchronous distributed admm for consensus optimization," in *Proceedings of the International Conference on Machine Learning*, 2014, pp. 1701–1709.
[11] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Proceedings of the Neural Information Processing Systems*, 2015, pp. 2737–2745.
[12] J. Liu and S. J. Wright, "Asynchronous stochastic coordinate descent: Parallelism and convergence properties," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 351–376, 2015.
[13] T.-H. Chang, M. Hong, W. Liao, and X. Wang, "Asynchronous distributed admm for large-scale optimization part I: Algorithm and convergence analysis," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3118–3130, 2016.
[14] T.-H. Chang, W. Liao, M. Hong, and X. Wang, "Asynchronous distributed admm for large-scale optimization part II: Linear convergence analysis and numerical performance," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3131–3144, 2016.
[15] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *Proceedings of the Neural Information Processing Systems*, 2011, pp. 873–881.
[16] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson, "An asynchronous mini-batch algorithm for regularized stochastic optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3740–3754, 2016.
[17] M. Hong and T.-H. Chang, "Stochastic proximal gradient consensus over random networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 11, pp. 2933–2948, 2017.
[18] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.
[19] C. Zhang, "Nearly unbiased variable selection under minimax concave penalty," *Annals of Statistics*, vol. 38, no. 2, pp. 894–942, 2010.
[20] Z. Xu, X. Chang, F. Xu, and H. Zhang, "$l_{1/2}$ regularization: A thresholding representation theory and a fast solver," *IEEE Transactions on Neural Networks*, vol. 23, no. 7, pp. 1013–1027, 2012.
[21] F. R. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Optimization with sparsity-inducing penalties," *Foundations and Trends in Machine Learning*, vol. 4, no. 1, pp. 1–106, 2012.
[22] Z. Allen-Zhu and E. Hazan, "Variance reduction for faster non-convex optimization," in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 699–707.
[23] E. Hazan, K. Singh, and C. Zhang, "Efficient regret minimization in non-convex games," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 1433–1441.
[24] Z. Allen-Zhu, "Natasha: Faster non-convex stochastic optimization via strongly non-convex parameter," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 89–97.
[25] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," in *Proceedings of the International Conference on Learning Representations*, 2018.
[26] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995.
[27] Q. Meng, W. Chen, J. Yu, T. Wang, Z. Ma, and T. Liu, "Asynchronous stochastic proximal optimization algorithms with variance reduction," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 2329–2335.

[28] J.-S. Pan, L.-G. Zhang, R.-B. Wang, V. Snášel, and S.-C. Chu, "Gannet optimization algorithm: A new metaheuristic algorithm for solving engineering optimization problems," *Mathematics and Computers in Simulation*, vol. 202, pp. 343–373, 2022.
[29] P. Hu, J.-S. Pan, S.-C. Chu, and C. Sun, "Multi-surrogate assisted binary particle swarm optimization algorithm and its application for feature selection," *Applied Soft Computing*, vol. 121, p. 108736, 2022.
[30] J.-S. Pan, Z. Zhang, S.-C. Chu, S.-Q. Zhang, and J. M.-T. Wu, "A parallel compact marine predators algorithm applied in time series prediction of backpropagation neural network (bnn) and engineering optimization," *Mathematics and Computers in Simulation*, vol. 220, pp. 65–88, 2024.
[31] T. Zhang, "Analysis of multi-stage convex relaxation for sparse regularization," *Journal of Machine Learning Research*, vol. 11, pp. 1081–1107, 2010.
[32] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B*, vol. 58, no. 1, pp. 267–288, 1996.
[33] E. J. Candès, J. K. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.
[34] R. Chartrand and V. Staneva, "Restricted isometry properties and nonconvex compressive sensing," *Inverse Problems*, vol. 24, no. 3, p. 035020, 2008.
[35] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," in *Proceedings of the International Conference on Machine Learning*, 2012, pp. 1571–1578.
[36] S. Ghadimi and G. Lan, "Stochastic first- and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.
[37] S. Ghadimi, G. Lan, and H. Zhang, "Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization," *Mathematical Programming*, vol. 155, pp. 267–305, 2016.
[38] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean, "Boosting algorithms as gradient descent in function space," in *Proceedings of the Neural Information Processing Systems*, 1999, pp. 512–518.
[39] J. Mairal, F. R. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the International Conference on Machine Learning*, 2009, pp. 689–696.
[40] J. Lin, L. Rosasco, and D. Zhou, "Iterative regularization for learning with convex loss functions," *Journal of Machine Learning Research*, vol. 17, no. 77, pp. 1–38, 2016.
[41] C. Fang and Z. Lin, "Parallel asynchronous stochastic variance reduction for nonconvex optimization," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 794–800.
[42] U. **c**Sim**c**sekli, C. Yıldız, T. H. Nguyen, G. Richard, and A. T. Cemgil, "Asynchronous stochastic quasi-newton MCMC for non-convex optimization," in *Proceedings of the International Conference on Machine Learning*, 2018.
[43] O. Shamir, "An optimal algorithm for bandit and zero-order convex optimization with two-point feedback," *Journal of Machine Learning Research*, vol. 18, no. 52, pp. 1–11, 2017.
[44] S. J. Reddi, S. Sra, B. Póczos, and A. J. Smola, "Fast incremental method for nonconvex optimization," in *Proceedings of the Conference on Decision and Control*, 2016, pp. 1971– 1977.
[45] M. Raginsky, A. Rakhlin, and M. Telgarsky, "Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis," *in Proceedings of Conference on Learning Theory*, pp. 1674–1703, 2017.
[46] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.
[47] O. Shamir and L. Szlak, "Online learning with local permutations and delayed feedback," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 3086–3094.

7. **Appendix.**

7.1. **Proof of Theorem 3.1.**

Proof. For simplicity, let $\Psi_k(x) := h(x_k) + \langle H_k, x - x_k\rangle + \frac{1}{2\eta_k}\|x - x_k\|^2 + r(x)$ and $\triangle_{x_k} := x_{k+1} - x_k$. Then, by the β-strong convexity of $\Psi_k(\cdot)$, we have

$$\begin{aligned}\frac{\beta}{2}\|\triangle_{x_k}\|^2 \leq& \Psi_k(x_k) - \Psi_k(x_{k+1}) - \langle\widetilde{\nabla}_{k+1}, x_k - x_{k+1}\rangle\\ =& \Psi_k(x_k) - \Psi_k(x_{k+1}),\end{aligned} \tag{13}$$

where $\widetilde{\nabla}_{k+1} := \widetilde{\nabla}\Psi_k(x_{k+1}) + \widetilde{\nabla} I_X(x_{k+1})$, $\widetilde{\nabla}\Psi_k(x_{k+1}) \in \partial\Psi_k(x_{k+1})$, $\widetilde{\nabla} I_X(x_{k+1}) \in \partial I_X(x_{k+1})$ and $I_X(x)$ is the indicate function, i.e., $I_X(x) = 0$ if $x \in X$ and $I_X(x) = +\infty$ otherwise. Here we use the fact that, by the optimality of x_{k+1}, there is a $\widetilde{\nabla}_{k+1}$ such that $\widetilde{\nabla}_{k+1} = 0$.

Let $\triangle_{\Psi_k(x)} := \Psi_{k+1}(x) - \Psi_k(x)$. Since $h \in \mathcal{C}_{L_h}(X)$, for any $k \in [N]$, we have

$$\begin{aligned}\triangle_{\Psi_k(x_{k+1})} =& h(x_{k+1}) - h(x_k) - \langle H_k, \triangle_{x_k}\rangle - \frac{1}{2\eta_k}\|\triangle_{x_k}\|^2\\ \leq& \frac{1}{m_k}\sum_{i=1}^{m_k}\langle\nabla h(x_{k-\tau_{k,i}};\xi_{k,i}) - \nabla h(x_k), -\triangle_{x_k}\rangle - \left(\frac{1}{2\eta_k} - \frac{L_h}{2}\right)\|\triangle_{x_k}\|^2.\end{aligned} \tag{14}$$

Further, for simplicity, let

$$\begin{aligned}&\frac{1}{m_k}\sum_{i=1}^{m_k}\langle\nabla h(x_{k-\tau_{k,i}};\xi_{k,i}) - \nabla h(x_k), \triangle_{x_k}\rangle\\ =&\underbrace{\frac{1}{m_k}\sum_{i=1}^{m_k}\langle\nabla h(x_{k-\tau_{k,i}};\xi_{k,i}) - \nabla h(x_{k-\tau_{k,i}}), -\triangle_{x_k}\rangle}_{I}\\ &+\underbrace{\frac{1}{m_k}\sum_{i=1}^{m_k}\langle\nabla h(x_{k-\tau_{k,i}}) - \nabla h(x_k), -\triangle_{x_k}\rangle}_{II}.\end{aligned}$$

For the term II, using the fact that $\langle a, b\rangle \leq \frac{1}{2}\|a\|^2 + \frac{1}{2}\|b\|^2$ and that $\|\sum_{i=1}^m a_i\|^2 \leq m\sum_{i=1}^m \|a_i\|^2$, it holds that

$$\begin{aligned}II \leq& \frac{1}{2m_k}\sum_{i=1}^{m_k}\left\|\nabla h(x_{k-\tau_{k,i}}) - \nabla h(x_k)\right\|^2 + \frac{1}{2}\|\triangle_{x_k}\|^2\\ \leq& \frac{L_h^2}{2m_k}\sum_{i=1}^{m_k}\|x_{k-\tau_{k,i}} - x_k\|^2 + \frac{1}{2}\|\triangle_{x_k}\|^2\\ \leq& \frac{L_h^2}{2m_k}\sum_{i=1}^{m_k}\tau_{k,i}\sum_{j=1}^{\tau_{k,i}}\|x_{k-\tau_{k,i}+j-1} - x_{k-\tau_{k,i}+j}\|^2 + \frac{1}{2}\|\triangle_{x_k}\|^2.\end{aligned} \tag{15}$$

To bound the term I, first we let $\widetilde{x}_{k+1} := \arg\min_{x\in X}\widetilde{\Psi}_k(x)$, where

$$\widetilde{\Psi}_k(x) := \left\{h(x_k) + \langle\widetilde{H}_k, x - x_k\rangle + \frac{1}{2\eta_k}\|x - x_k\|^2 + r(x)\right\},$$

and $\widetilde{H}_k = \frac{1}{m_k}\sum_{i=1}^{m_k}\nabla h(x_{k-\tau_{k,i}})$. Further, let $\triangle_{\widetilde{x}_{k+1}} := \widetilde{x}_{k+1} - x_{k+1}$ and $\delta_{k,i} := \nabla h(x_{k-\tau_{k,i}};\xi_{k,i}) - \nabla h(x_{k-\tau_{k,i}})$. Then, by the β-strong convexity of $\Psi_k(\cdot)$, we have

$$\begin{aligned}\frac{\beta}{2}\|\triangle_{\widetilde{x}_{k+1}}\|^2 \leq& \Psi_k(\widetilde{x}_{k+1}) - \Psi_k(x_{k+1}) - \langle\widetilde{\nabla}_{k+1}, \triangle_{\widetilde{x}_{k+1}}\rangle \\ =& \widetilde{\Psi}_k(\widetilde{x}_{k+1}) + \frac{1}{m_k}\sum_{i=1}^{m_k}\langle\delta_{k,i}, \triangle_{\widetilde{x}_{k+1}}\rangle - \widetilde{\Psi}_k(x_{k+1}).\end{aligned}$$

Further, by the fact that $\widetilde{\Psi}_k(\widetilde{x}_{k+1}) \leq \widetilde{\Psi}_k(x_{k+1})$, we have

$$\frac{\beta}{2}\|\widetilde{x}_{k+1} - x_{k+1}\| \leq \left\|\frac{1}{m_k}\sum_{i=1}^{m_k}\delta_{k,i}\right\|.$$

Then, it holds that

$$\begin{aligned}I =& \frac{1}{m_k}\sum_{i=1}^{m_k}[\langle\delta_{k,i}, x_k - \widetilde{x}_{k+1}\rangle + \langle\delta_{k,i}, \widetilde{x}_{k+1} - x_{k+1}\rangle] \\ \leq& \frac{1}{m_k}\sum_{i=1}^{m_k}\langle\delta_{k,i}, x_k - \widetilde{x}_{k+1}\rangle + \frac{2}{\beta}\left\|\frac{1}{m_k}\sum_{i=1}^{m_k}\delta_{k,i}\right\|^2.\end{aligned} \tag{16}$$

Using the fact that $\triangle_{\Phi(x_k)} := \Phi(x_k) - \Phi(x_{k+1}) = [\Psi_k(x_k) - \Psi_k(x_{k+1})] - [\Psi_{k+1}(x_{k+1}) - \Psi_k(x_{k+1})]$ and combining (13), (14), (15) and (16) together, we have

$$\begin{aligned}\triangle_{\Phi(x_k)} \geq& \frac{1}{m_k}\sum_{i=1}^{m_k}\langle\delta_{k,i}, \widetilde{x}_{k+1-x_k}\rangle - \frac{2}{\beta}\left\|\frac{1}{m_k}\sum_{i=1}^{m_k}\delta_{k,i}\right\|^2 \\ &- \frac{L_h^2}{2m_k}\sum_{i=1}^{m_k}\tau_{k,i}\sum_{j=1}^{\tau_{k,i}}\|x_{k-\tau_{k,i}+j-1} - x_{k-\tau_{k,i}+j}\|^2 \\ &+ \left(\frac{1}{2\eta_k} - \frac{L_h}{2} + \frac{\beta}{2} - \frac{1}{2}\right)\|\triangle_{x_k}\|^2.\end{aligned} \tag{17}$$

On the other hand, it holds that

$$\begin{aligned}&\mathbb{E}_{\xi_{[k]}}\left[\left\|\sum_{i=1}^{m_k}\delta_{k,i}\right\|^2\right] \\ =& \mathbb{E}_{\xi_{[k]}}\left[\left\|\sum_{i=1}^{m_k-1}\delta_{k,i}\right\|^2 + 2\left\langle\sum_{i=1}^{m_k-1}\delta_{k,i}, \delta_{k,m_k}\right\rangle + \|\delta_{k,m_k}\|^2\right] \\ =& \mathbb{E}_{\xi_{[k]}}\left[\left\|\sum_{i=1}^{m_k-1}\delta_{k,i}\right\|^2\right] + \mathbb{E}_{\xi_{[k]}}\left[\|\delta_{k,m_k}\|^2\right],\end{aligned}$$

which means, following Assumption (**A**2), that

$$\mathbb{E}_{\xi_{[k]}}\left[\left\|\frac{1}{m_k}\sum_{i=1}^{m_k}\delta_{k,i}\right\|^2\right] = \frac{1}{m_k^2}\sum_{i=1}^{m_k}\mathbb{E}_{\xi_{[k]}}\left[\|\delta_{k,i}\|^2\right] \leq \frac{\sigma^2}{m_k}.$$

Rolling up both sides of (17) over $k \in [N]$, we have

$$\begin{aligned}\sum_{k=1}^{N} \triangle_{\Phi(x_k)} \geq & \sum_{k=1}^{N} \frac{1}{m_k} \sum_{i=1}^{m_k} \langle \delta_{k,i}, \widetilde{x}_{k+1} - x_k \rangle - \frac{2\sigma^2}{\beta} \sum_{k=1}^{N} \frac{1}{m_k} - \frac{L_h^2 \tau}{2} \sum_{k=1}^{N} \sum_{j=1}^{\tau} \|\triangle_{x_{k-\tau+j}}\|^2 \\ & + \sum_{k=1}^{N} \left(\frac{1}{2\eta_k} - \frac{L_h}{2} + \frac{\beta}{2} - \frac{1}{2} \right) \|\triangle_{x_k}\|^2 \\ \geq & \sum_{k=1}^{N} \frac{1}{m_k} \sum_{i=1}^{m_k} \langle \delta_{k,i}, \widetilde{x}_{k+1} - x_k \rangle - \frac{2\sigma^2}{\beta} \sum_{k=1}^{N} \frac{1}{m_k} \\ & + \sum_{k=1}^{N} \left(\frac{1}{2\eta_k} - \frac{L_h}{2} + \frac{\beta}{2} - \frac{1}{2} - \frac{L_h^2 \tau^2}{2} \right) \|\triangle_{x_k}\|^2,\end{aligned} \tag{18}$$

where $\triangle_{x_{k-\tau+j}} := x_{k-\tau+j-1} - x_{k-\tau+j}$ and we let $x_k \equiv 0$ for any $k < 0$. In the first inequality we used the fact that

$$\frac{L_h^2}{2m_k} \sum_{i=1}^{m_k} \tau_{k,i} \sum_{j=1}^{\tau_{k,i}} \|\triangle_{x_{k-\tau_{k,i}+j}}\|^2 \leq \frac{L_h^2 \tau}{2} \sum_{j=1}^{\tau} \|\triangle_{x_{k-\tau+j}}\|^2,$$

where $\triangle_{x_{k-\tau_{k,i}+j}} := x_{k-\tau_{k,i}+j-1} - x_{k-\tau_{k,i}+j}$, and in the second inequality the fact that

$$\sum_{k=1}^{N} \sum_{j=1}^{\tau} \|\triangle_{x_{k-\tau+j}}\|^2 \leq \tau \sum_{k=1}^{N} \|\triangle_{x_k}\|^2.$$

Now take expectation w.r.t. $\xi_{[N]}$ on both sides of (18) and we get

$$\sum_{k=1}^{N} \left(\frac{1}{2\eta_k} - \frac{L_h}{2} + \frac{\beta}{2} - \frac{1}{2} - \frac{L_h^2 \tau^2}{2} \right) \mathbb{E}_{\xi_{[k]}} \left[\|\triangle_{x_k}\|^2 \right] \leq \Phi(x_1) - \Phi^* + \frac{2\sigma^2}{\beta} \sum_{k=1}^{N} \frac{1}{m_k}, \tag{19}$$

where we used Assumption (**A**1), i.e., $\mathbb{E}_{\xi_{[k]}}[\langle \delta_{k,i}, \widetilde{x}_{k+1} - x_k \rangle] = \mathbb{E}_{\xi_{[k-1]}} \mathbb{E}_{\xi_k}[\langle \delta_{k,i}, \widetilde{x}_{k+1} - x_k \rangle | \xi_{[k-1]}] = 0$ and the fact that $\Phi(x_{N+1}) \geq \Phi^* := \min_{x \in X} \Phi(x)$. Further, by the definition of the projected gradient (4), we can replace $\|\triangle_{x_k}\|^2$ with $\eta_k^2 \|g_X(x_k, H_k, \eta_k)\|^2$ in (19).

It can be verified that $\sum_{k=1}^{N} \left(\frac{\eta_k}{2} - \frac{(\tau^2 L_h^2 + L_h + 1 - \beta)\eta_k^2}{2} \right)$ is positive.

Then, dividing both sides of (19) by this term and using the fact that

$$\mathbb{E}_{R, \xi_{[N]}} \left[\|g_X(x_R, H_R, \eta_R)\|^2 \right] = \sum_{k=1}^{N} \frac{\left[\eta_k - (\tau^2 L_h^2 + L_h + 1 - \beta)\eta_k^2 \right] \mathbb{E}_{\xi_{[k]}} \|g_X(x_k, H_k, \eta_k)\|^2}{\sum_{j=1}^{N} \left[\eta_j - (\tau^2 L_h^2 + L_h + 1 - \beta)\eta_j^2 \right]},$$

the conclusion (a) is yielded.

In Case II where both $h(\cdot)$ and $r(\cdot)$ are convex, Ψ_k is at least $\frac{1}{\eta_k}$-strongly convex. Then, for any $x \in X$, we have

$$\Psi_k(x) - \Psi_k(x_{k+1}) - \langle \widetilde{\nabla}_{k+1}, x - x_{k+1} \rangle \geq \frac{1}{2\eta_k} \|x - x_{k+1}\|^2.$$

Further, by the definition of $\Psi_k(\cdot)$, it can be rewritten as

$$\frac{1}{2\eta_k} \|\triangle_{x_k}\|^2 + r(x_{k+1}) \leq \langle H_k, x - x_{k+1} \rangle + r(x) + \frac{1}{2\eta_k} (\|x - x_k\|^2 - \|x - x_{k+1}\|^2), \tag{20}$$

where we used the fact that $\widetilde{\nabla}_{k+1} = 0$. Let $\triangle_{x_{k-\tau_{k,i}}} := x_{k+1} - x_{k-\tau_{k,i}}$ for short. Since h is convex and $h \in \mathcal{C}_{L_h}(X)$, we have

$$\begin{aligned} h(x_{k+1}) \leq & h(x_{k-\tau_{k,i}}) + \langle \nabla h(x_{k-\tau_{k,i}}), \triangle_{x_{k-\tau_{k,i}}} \rangle + \frac{L_h}{2} \|\triangle_{x_{k-\tau_{k,i}}}\|^2 \\ \leq & h(x) + \langle \nabla h(x_{k-\tau_{k,i}}), x_{k+1} - x \rangle + \frac{L_h}{2} \|\triangle_{x_{k-\tau_{k,i}}}\|^2. \end{aligned}$$

Then, $m_k h(x_{k+1})$ is upper bounded by

$$m_k h(x) + \sum_{i=1}^{m_k} \langle \nabla h(x_{k-\tau_{k,i}}), x_{k+1} - x \rangle + \frac{L_h}{2} \sum_{i=1}^{m_k} \|\triangle_{x_{k-\tau_{k,i}}}\|^2.$$

Consequently, by the definition of H_k and $\delta_{k,i}$, we have

$$\begin{aligned} \langle H_k, x - x_{k+1} \rangle \leq & h(x) - h(x_{k+1}) + \frac{1}{m_k} \sum_{i=1}^{m_k} \langle \delta_{k,i}, x - x_{k+1} \rangle + \frac{L_h}{2m_k} \sum_{i=1}^{m_k} \|\triangle_{x_{k-\tau_{k,i}}}\|^2 \\ \leq & h(x) - h(x_{k+1}) + \left\| \frac{1}{m_k} \sum_{i=1}^{m_k} \delta_{k,i} \right\| \cdot \|\triangle_{x_k}\| + \frac{1}{m_k} \sum_{i=1}^{m_k} \langle \delta_{k,i}, x - x_k \rangle \\ & + \frac{L_h}{2m_k} \sum_{i=1}^{m_k} (\tau_{k,i} + 1) \sum_{j=1}^{\tau_{k,i}+1} \|\triangle_{x_{k-\tau_{k,i}+j}}\|^2, \end{aligned} \tag{21}$$

where we used the fact in the second inequality that

$$\sum_{i=1}^{m_k} \|\triangle_{x_{k-\tau_{k,i}}}\|^2 \leq \sum_{i=1}^{m_k} (\tau_{k,i} + 1) \sum_{j=1}^{\tau_{k,i}+1} \|\triangle_{x_{k-\tau_{k,i}+j}}\|^2.$$

Notice that, (20) and (21) are true for any $x \in X$, including $x^* = \arg\min_{x \in X} \Phi(x)$. Then, by substituting (21) into (20), and rolling up its both sides over $k \in [N]$, we have

$$\begin{aligned} \sum_{k=1}^{N} \Phi(x_{k+1}) - \Phi^* \leq & \sum_{k=1}^{N} \left\| \frac{1}{m_k} \sum_{i=1}^{m_k} \delta_{k,i} \right\| \cdot \|\triangle_{x_k}\| + \sum_{k=1}^{N} \frac{1}{m_k} \sum_{i=1}^{m_k} \langle \delta_{k,i}, x^* - x_k \rangle \\ & + \frac{1}{2\eta_1} \|x^* - x_1\|^2 - \frac{1}{2\eta_N} \|x^* - x_{N+1}\|^2 - \sum_{k=1}^{N} \left(\frac{1}{2\eta_k} - \frac{L_h(\tau+1)^2}{2} \right) \|\triangle_{x_k}\|^2, \end{aligned} \tag{22}$$

where we let $x_k \equiv 0$ for any $k < 0$ and used the fact that

$$\sum_{k=1}^{N} \frac{1}{m_k} \sum_{i=1}^{m_k} \sum_{j=1}^{\tau_{k,i}+1} \|\triangle_{x_{k-\tau_{k,i}+j}}\|^2 \leq (\tau+1) \sum_{k=1}^{N} \|\triangle_{x_k}\|^2.$$

Further, noticing that $\forall a, b \in \mathbb{R}$ and $b > 0$, $ax - \frac{bx^2}{2} \leq \frac{a^2}{2b}$, we have

$$\begin{aligned} & \sum_{k=1}^{N} \left\| \frac{1}{m_k} \sum_{i=1}^{m_k} \delta_{k,i} \right\| \cdot \|\triangle_{x_k}\| - \sum_{k=1}^{N} \left(\frac{1}{2\eta_k} - \frac{L_h(\tau+1)^2}{2} \right) \|\triangle_{x_k}\|^2 \\ & \qquad \leq \sum_{k=1}^{N} \frac{1}{2[1/\eta_k - L_h(\tau+1)^2]} \left\| \frac{1}{m_k} \sum_{i=1}^{m_k} \delta_{k,i} \right\|^2. \end{aligned}$$

Now, we take expectation on both sides of (22) w.r.t. $\xi_{[N]}$ and we have

$$\sum_{k=1}^{N}\mathbb{E}_{\xi_{[k]}}[\Phi(x_{k+1})-\Phi^*]\leq\sum_{k=1}^{N}\frac{\sigma^2}{2[1/\eta_k-L_h(\tau+1)^2]m_k}+\frac{1}{2\eta_1}\|x^*-x_1\|^2. \tag{23}$$

where we used Assumption (**A**1), i.e., $\mathbb{E}_{\xi_{[k]}}[\langle\delta_{k,i},x^*-x_k\rangle]=0$ and Assumption (**A**2). Finally, notice the fact that $\mathbb{E}_{R,\xi_{[N]}}[\Phi(x_R)-\Phi^*]=\frac{1}{N}\sum_{k=1}^{N}\mathbb{E}_{\xi_{[k]}}[\Phi(x_k)-\Phi^*]$, then the conclusion (b) follows from (23). □

7.2. **Proof of Theorem 4.1.**

Proof. First, let $\Phi_\mu(x):=h_\mu(x)+r(x)$. From the claim (a) of Lemma 7.1, we know that $\mathbb{E}_{\zeta,\xi}[\nabla h_\mu(x;\xi,\zeta)]=\nabla h_\mu(x)$. To verify the claim (a) of Theorem 4.1, we only need to ensure that the variance of the approximate stochatic gradient is bounded for any $k\in[N]$ and $i\in[m_k]$. For simplicity, the subscripts k and i are omitted here. By a direct calculation, we have

$$\begin{aligned}\mathbb{E}_{\xi,\zeta}\left[\|\nabla h_\mu(x;\xi,\zeta)-\nabla h_\mu(x)\|^2\right]=&\mathbb{E}_{\xi,\zeta}\left[\|\nabla h_\mu(x;\xi,\zeta)\|^2\right]-2\mathbb{E}_{\xi,\zeta}\left[\langle\nabla h_\mu(x;\xi,\zeta),\nabla h_\mu(x)\rangle\right]\\&+\|\nabla h_\mu(x)\|^2\\=&\mathbb{E}_{\xi,\zeta}\left[\|\nabla h_\mu(x;\xi,\zeta)\|^2\right]-\|\nabla h_\mu(x)\|^2.\end{aligned}$$

Further, for the first term, we have

$$\begin{aligned}\mathbb{E}_{\xi,\zeta}\left[\|\nabla h_\mu(x;\xi,\zeta)\|^2\right]=&\mathbb{E}_{\xi}\left[\mathbb{E}_{\zeta}\left[\|\nabla h_\mu(x;\xi,\zeta)\|^2\right]\right]\\\leq&2(d+4)\mathbb{E}_{\xi}\left[\|\nabla h_\mu(x;\xi)\|^2\right]+\frac{\mu^2}{2}L_h^2(d+6)^2\\\leq&2(d+4)(C^2+\sigma^2)+\frac{\mu^2}{2}L_h^2(d+6)^2,\end{aligned}$$

where the claim (b) of Lemma 7.1 is used in the first inequality. Therefore, $\nabla h_\mu(x_k;\xi_k,\zeta)$ satisfies the assumptions (**A**1) and (**A**2), and the conclusions of Theorem 3.1 is applicable for ARSPGF algorithm.

In particular, using directly the claim (a) of Theorem 3.1, the claim (a) of Theorem 4.1 is yielded. For the claim (b), using the claim (b) of Theorem 3.1, we have

$$\mathbb{E}[\Phi_\mu(x_R)-\Phi_\mu^*]\leq\frac{\sum_{k=1}^{N}\frac{\widetilde{\sigma}^2}{2[1/\eta_k-L_h(\tau+1)^2]m_k}+\frac{1}{2\eta_1}\|x^*-x_1\|^2}{N},$$

where $\Phi_\mu^*:=\min_{x\in X}\Phi_\mu^*(x)$. Notice that we aim to find the optimum solution of Φ rather than that of Φ_μ. Following the conclusion (b) of Lemma 7.1, it holds that

$$|\Phi_\mu(x_R)-\Phi_\mu^*-(\Phi(x_R)-\Phi^*)|\leq\mu^2dL_h.$$

Then the claim (b) is yielded. □

7.3. **Restate Lemma 5 in** [46]**.**

Lemma 7.1. [46] *If $h\in\mathcal{C}_{L_h}(X)$, then*
(a) $h_\mu\in\mathcal{C}_{L_{h_\mu}}(X)$ where $L_{h_\mu}\leq L_h$, and

$$\nabla h_\mu(x)=\frac{1}{(2\pi)^{n/2}}\int\frac{h(x+\mu\zeta)-h(x)}{\mu}\zeta\exp\left(-\frac{\|\zeta\|^2}{2}\right)d\zeta.$$

(b) for any $x \in \mathbb{R}^d$, we have

$$i)\ |h_\mu(x) - h(x)| \leq \frac{\mu^2}{2} d L_h,$$

$$ii)\ \|\nabla h_\mu(x) - \nabla h(x)\| \leq \frac{\mu^2}{2} L_h (d+3)^{3/2},$$

$$iii)\ \mathbb{E}_\zeta \left[\left\| \frac{h(x+\mu\zeta) - h(x)}{\mu} \zeta \right\|^2 \right] \leq 2(d+4)\|\nabla h(x)\|^2 + \frac{\mu^2}{2} L_h^2 (d+6)^3.$$

(c) h_μ is also convex provided h is convex.