Dimension-free Complexity Bounds for High-order Nonconvex Finite-sum Optimization

Dongruo Zhou [1] **Quanquan Gu** [1]

Abstract

Stochastic high-order methods for finding first-order stationary points in nonconvex finite-sum optimization have witnessed increasing interest in recent years, and various upper and lower bounds of the oracle complexity have been proved. However, under standard regularity assumptions, existing complexity bounds are all *dimension-dependent* (e.g., polylogarithmic dependence), which contrasts with the dimension-free complexity bounds for stochastic first-order methods and deterministic high-order methods. In this paper, we show that the polylogarithmic dimension dependence gap is not essential and can be closed. More specifically, we propose stochastic high-order algorithms with novel first-order and high-order derivative estimators, which can achieve dimension-free complexity bounds. With the access to p-th order derivatives of the objective function, we prove that our algorithm finds ϵ-stationary points with $O(n^{(2p-1)/(2p)}/\epsilon^{(p+1)/p})$ high-order oracle complexities, where n is the number of individual functions. Our result strictly improves the complexity bounds of existing high-order deterministic methods with respect to the dependence on n, and it is *dimension-free* compared with existing stochastic high-order methods.

1. Introduction

We study the following nonconvex finite-sum optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}), \tag{1.1}$$

where n is the number of individual functions and each $f_i : \mathbb{R}^d \to \mathbb{R}$ can possibly be a nonconvex function. Due to the lack of convexity, it can, in the worst case, be NP-hard to find the global minimum of (1.1) for some specific function f (Hillar & Lim, 2013). Thus, our goal is to instead find an ϵ-stationary point $\mathbf{x}$, which is defined by

$$\|\nabla f(\mathbf{x})\|_2 \le \epsilon.$$

Understanding the complexity of finding stationary points for nonconvex finite-sum optimization has been a central problem in both the machine learning and the optimization communities. Gradient descent, which is probably the most basic algorithm for the above goal, can find ϵ-stationary points within $O(\epsilon^{-2})$ number of iterations, or equivalently, $O(n\epsilon^{-2})$ number of gradient evaluations of the individual functions. Starting from gradient descent and its complexity, numerous algorithms have been proposed with improvement from different aspects. The improvement can be mainly summarized into three categories:

- *Better dependence on ϵ with the help of high-order information*. By only gradient information, it is well-known that $O(\epsilon^{-2})$ is the optimal iteration complexity (Nesterov, 2013). To break the ϵ^{-2} barrier, high-order information including Hessian and higher-order derivatives have to be employed, and better dependence on ϵ^{-1} can indeed be achieved. Representative algorithms include cubic regularization (Nesterov & Polyak, 2006) with $O(\epsilon^{-3/2})$ iteration complexity and high-order regularized method (HR) (Birgin et al., 2017) with $O(\epsilon^{-(p+1)/p})$ iteration complexity, where p is the order of derivatives being used.
- *Better dependence on n with the variance-reduction technique*. To improve the dependence on n, variance reduction technique is often employed. More specifically, a semi-stochastic gradient estimator called variance-reduced gradient has been firstly proposed for finite-sum convex optimization (Roux et al., 2012; Johnson & Zhang, 2013) with a better gradient complexity. Latter on, a recursive (Nguyen et al., 2017; Fang et al., 2018) and a nested gradient estimator (Zhou et al., 2018c) have been

[1]Department of Computer Science, University of California, Los Angeles, CA 90095, USA. Correspondence to: Quanquan Gu <qgu@cs.ucla.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Table 1. Comparisons of different methods to find ϵ-stationary points w.r.t. their oracle complexity

Algorithm	Complexity	Lipschitiz continuity assumption	Dimension-free?	Small batch size?
CR (Nesterov & Polyak, 2006)	$O\big(\frac{n}{\epsilon^{3/2}}\big)$	Second-order	yes	no
SVRC (Zhou et al., 2018a)	$O\big(\log d \cdot \frac{n^{4/5}}{\epsilon^{3/2}}\big)$	Second-order	no	no
STR2 (Shen et al., 2019)	$O\big(\log d \cdot \frac{n^{3/4}}{\epsilon^{3/2}}\big)$	Second-order	no	no
HR (Birgin et al., 2017)	$O\big(\frac{n}{\epsilon^{(p+1)/p}}\big)$	p-th-order	yes	no
OP-TE (Algorithm 2)	$O\big(\frac{n^{(3p-1)/(3p)}}{\epsilon^{(p+1)/p}}\big)$	p-th-order	**yes**	no
TP-TE (Algorithm 3)	$O\big(\frac{n^{(2p-1)/(2p)}}{\epsilon^{(p+1)/p}}\big)$	p-th-order	**yes**	**yes**
Lower bound (Emmenegger et al., 2021)	$\Omega\big(\frac{n^{(p-1)/(2p)}}{\epsilon^{(p+1)/p}}\big)$	p-th-order	–	–

proposed for the nonconvex setting with $O(n^{1/2}\epsilon^{-2})$ gradient complexity, which strictly improves that of gradient descent by an $O(n^{1/2})$ factor.

- *Better dependence on n and ϵ at the price of dimension dependence*. This line of works improve the dependence both on n and ϵ by introducing variance-reduced gradient and high-order derivative estimators. For instance, under the standard Hessian Lipschitz assumption, Zhou et al. (2018a) proposed an SVRC algorithm with $O(\log d \cdot n^{4/5}/\epsilon^{3/2})$ number of second-order oracle calls. Shen et al. (2019) proposed an STR algorithm with $O(\log d \cdot n^{3/4}/\epsilon^{3/2})$ number of second-order oracle calls. Due to the use of semi-stochastic Hessian, both of these works have a logarithmic dependence on dimension d, which makes them not fully dimension-free, unlike gradient methods.[1]

Given these existing works, it is natural to ask:

Is it possible to design a dimension-free *algorithm with better dependence on n and ϵ?*

In this work, we answer this question affirmatively by proposing two algorithms, Single-Point Taylor Expansion (OP-TE) and Two-Point Taylor Expansion (TP-TE). Both of these two algorithms utilize p-th order information of function f by constructing a stochastic Taylor series-based derivative estimators. To find ϵ-stationary points, we show that OP-TE enjoys an $O(n^{(3p-1)/(3p)}/\epsilon^{(p+1)/p})$ oracle complexity and TP-TE enjoys an $O(n^{(2p-1)/(2p)}/\epsilon^{(p+1)/p})$ oracle complexity. It is worth noting that both complexity results are independent of the *dimension*, and unlike previous approaches (Kohler & Lucchi, 2017; Zhou et al., 2018a), the result of TP-TE can be attained without choosing a large batch size (i.e., the batch size can be chosen as 1). Our result is the first of its kind in stochastic high-order optimization that is dimension-free and supports small batch size, while achieving the state-of-the-art oracle complexity. We compare our results with previous ones (including both upper and lower bounds) in Table 1.

Notation Let x_j denote the j-th entry of an vector $\mathbf{x}$ and $\|\mathbf{x}\|_2$ denotes its Euclidean norm. Let $[n]$ denote $\{1, \dots, n\}$. A d-dimensional p-th order tensor $\mathbf{J}$ is a multilinear operator $(\mathbb{R}^d)^p \to \mathbb{R}$, which is defined as

$$\mathbf{J}(\mathbf{v}_1, \dots, \mathbf{v}_p) := \sum_{i_1=1}^{d} \cdots \sum_{i_p=1}^{d} v_{i_1} \dots v_{i_p} J_{i_1,\dots,i_p}.$$

For any tensor $\mathbf{J}$, we denote its Euclidean operator norm as $\|\mathbf{J}\|_{\text{op}}$, which is

$$\|\mathbf{J}\|_{\text{op}} := \sup_{\|\mathbf{v}_1\|_2 \le 1, \dots, \|\mathbf{v}_p\|_2 \le 1} |\mathbf{J}(\mathbf{v}_1, \dots, \mathbf{v}_p)|.$$

For any two tensors $\mathbf{J}$ and $\mathbf{I}$, let $\langle \mathbf{J}, \mathbf{I} \rangle$ denote their inner product, which is

$$\langle \mathbf{J}, \mathbf{I} \rangle = \sum_{i_1=1}^{d} \cdots \sum_{i_p=1}^{d} J_{i_1,\dots,i_p} I_{i_1,\dots,i_p}.$$

Let $\otimes$ be the Kronecker product. For any vector $\mathbf{x}$, let $\mathbf{x}^{\otimes k}$ denote $\mathbf{x} \otimes \cdots \otimes \mathbf{x}$ for k times. Let $\mathbf{J}$ be a d-dimensional

[1] The only notable exception is the algorithm proposed by Zhang et al. (2018). However, they made a uncommon Hessian Lipschitz assumption in terms of the Frobenius norm rather than the standard counterpart in terms of the operator norm. By bounding the Frobenius norm with operator norm, their complexity result yields an even worse polynomial dependence on d.

p-th order tensor, $\mathbf{x} \in \mathbb{R}^d$, then for any $j \le p$, we use $\langle \mathbf{J}, \mathbf{x}^{\otimes j} \rangle$ to denote the $(p-j)$-th order tensor $\mathbf{I}$ such that for any $\mathbf{y} \in \mathbb{R}^d$,

$$\mathbf{I}(\mathbf{y}, \ldots, \mathbf{y}) = \langle \mathbf{J}, \mathbf{x}^{\otimes j} \otimes \mathbf{y}^{\otimes (p-j)} \rangle.$$

We call a tensor is symmetric if for any $j_1, \ldots, j_p$ which is a permutation of $i_1, \ldots, i_p$, we have $J_{j_1,\ldots,j_p} = J_{i_1,\ldots,i_p}$. In this work we only consider symmetric tensors. For a symmetric tensor $\mathbf{J}$, Zhang et al. (2012) showed that its operator norm can be defined by

$$\|\mathbf{J}\|_{\text{op}} = \sup_{\|\mathbf{v}\|_2=1} \langle \mathbf{J}, \mathbf{v}^{\otimes p} \rangle.$$

For any $p \ge 1$ and p-th order continuous differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, let $\nabla^p f(\mathbf{x})$ denote the tensor of p-th order partial derivates of f at $\mathbf{x}$, that is,

$$[\nabla^p f(\mathbf{x})]_{i_1,\ldots,i_p} = \frac{\partial^p f}{\partial x_{i_1} \cdots \partial x_{i_p}}(\mathbf{x}),\ i_1, \ldots, i_p \in [d].$$

$\nabla^p f(\mathbf{x})$ is symmetric due to its continuous differentiable property, that is, let $j_1, \ldots, j_p$ be a permutation of $i_1, \ldots, i_p$, we have $[\nabla^p f(\mathbf{x})]_{i_1,\ldots,i_p} = [\nabla^p f(\mathbf{x})]_{j_1,\ldots,j_p}$.

In this work, all index subsets are multiset. For any tensor $\mathbf{J}$ that can be written as $\mathbf{J} = 1/n \cdot \sum_{i=1}^n \mathbf{J}_i$, we use $\mathbf{J}_{\mathcal{I}}(\mathbf{x})$ to represent $1/|\mathcal{I}| \cdot \sum_{i \in \mathcal{I}} \mathbf{J}_i(\mathbf{x})$. We use $f_n = O(g_n)$ to denote that $f_n \le C g_n$ for some constant $C > 0$ and use $f_n = \widetilde{O}(g_n)$ to hide the logarithmic factors of g_n.

2. Related Work

There is a series of works studying p-order optimization with the special cases $p = 1$ or $p = 2$. Due to the space limit, we do not provide a comprehensive review here. Instead, we mainly review related works on p-th order ($p \ge 2$) optimization and variance reduction techniques. The following works focus on finding either ϵ-stationary points, (ϵ, ϵ_h)-second-order stationary points, or a q-th order (ϵ, δ)-approximate local minimizer, which is defined as a point $\mathbf{x}$ satisfying $\phi_j^\delta(\mathbf{x}) \le \epsilon \delta^j / j!$ for all $1 \le j \le q$, where $\phi_j^\delta(\mathbf{x})$ is the maximum value of the p-th order Taylor expansion of f expanded at $\mathbf{x}$. Such a notion is firstly proposed and studied by Cartis et al. (2016).

Stochastic second-order optimization The most related works are stochastic second-order optimization. The pioneer work Nesterov & Polyak (2006) proposed the deterministic cubic-regularized Newton method and proved that it finds $(\epsilon, \sqrt{\epsilon})$-second-order stationary points within $O(\epsilon^{-3/2})$ number of iterations. To extend it from deterministic case to stochastic case, Kohler & Lucchi (2017); Xu et al. (2017) proposed algorithms with subsampled gradient and subsampled Hessian within $\widetilde{O}(n\epsilon^{-3/2} \wedge \epsilon^{-7/2})$ gradient complexity and $\widetilde{O}(n\epsilon^{-3/2} \wedge \epsilon^{-5/2})$ Hessian complexity. Zhou et al. (2018a) proposed a stochastic variance reduced cubic regularization method (SVRC) for the finite-sum setting that attains $O(n^{4/5}\epsilon^{-3/2})$ second-order oracle complexity. Zhou et al. (2018b); Wang et al. (2018b); Zhang et al. (2018) used different gradient and Hessian estimators and obtain a better Hessian complexity, i.e., $O(n^{2/3}\epsilon^{-3/2})$. Our work fits in these works by considering the special case $p = 2$.

High-order optimization In recent years, high-order optimization has attracted most researchers' attention. For instance, Birgin et al. (2017) firstly proposed a regularized minimization method that finds stationary points within $O(\epsilon^{(p+1)/p})$ number of iterations. Cartis et al. (2020a) proposed an algorithm with a sharp $O(\epsilon^{(p+1)/(p-q+1)})$ number of function valuations to find approximate local minimizer. Cartis et al. (2017) proposed an adaptive regularized method that finds second-order stationary points within $O(\epsilon^{(p+1)/p} + \epsilon_h^{(p+1)/(p-1)})$ number of iterations. Cartis et al. (2020b) proposed deterministic but exact trust-region methods that find approximate local minimizer within $O(\epsilon^{-(q+1)})$ function evaluations. Bellavia et al. (2021) proposed stochastic trust-region methods that find approximate local minimizer within similar $O(\epsilon^{-(q+1)})$ number of function evaluations. Birgin et al. (2020) proposed a computational feasible fourth-order regularization method that finds stationary points within $O(\epsilon^{-4/3})$ number of iterations. Corresponding to above mentioned upper bound results, There are another of line of works providing lower bounds of complexity. For instance, Cartis et al. (2020a) provided a hard instance that shows their proposed algorithm needs at least $O(\epsilon^{(p+1)/(p-q+1)})$ number of function valuations to find approximate local minimizer. Carmon et al. (2017) constructed a hard instance that suggests any deterministic or randomized algorithm needs at least $O(\epsilon^{(p+1)/p})$ number of function evaluations to find stationary points. Arjevani et al. (2020) suggested that any stochastic algorithm with an access to the stochastic and Hessian-vector-product oracle needs at least $O(\epsilon^{-3})$ number of oracle calls to find stationary points, and $O(\epsilon^{-3} + \epsilon_h^{-5})$ number of oracle calls to find second-order stationary points. Our work fits in this line of works that studies the finite-sum setting with a best $O(n^{(2p-1)/(2p)}/\epsilon^{(p+1)/p})$ number of oracle calls.

Variance reduction Variance reduction technique is firstly proposed for first-order convex finite-sum optimization (Roux et al., 2012; Johnson & Zhang, 2013; Xiao & Zhang, 2014; Defazio et al., 2014; Nguyen et al., 2017). For nonconvex finite-sum optimization, to find stationary points, Reddi et al. (2016); Allen-Zhu & Hazan (2016) showed an $O(n^{2/3}/\epsilon^2)$ gradient complexity. Later on, the recursive/nested gradient estimators have been studied (Fang et al., 2018; Zhou et al., 2018c; Wang et al., 2018a; Nguyen et al., 2019; Cutkosky & Orabona, 2019) and further improved the gradient complexity to $O(n^{1/2}/\epsilon^2)$. Such a complexity has been shown to be near-optimal (Fang et al., 2018; Zhou & Gu, 2019). Our work extends this direction by proposing stochastic high-order Taylor expansion-based estimators.

3. Preliminaries and Assumptions

In this work, we assume our algorithms have access to the following *incremental high-order oracle (IHO)*, which has been introduced in Emmenegger et al. (2021).

Definition 3.1. Given a function $f = 1/n \cdot \sum_{i=1}^n f_i$, the incremental high-order oracle (IHO) $\mathcal{O}(\mathbf{x}, i)$ returns the following tuple of tensors:

$$\mathcal{O}(\mathbf{x}, i) := [\nabla f_i(\mathbf{x}), \nabla^2 f_i(\mathbf{x}), \dots, \nabla^p f_i(\mathbf{x})].$$

Clearly, IHO is the extension of the existing oracles *incremental first-order oracle* (Agarwal & Bottou, 2015) and *second-order oracle* (Zhou et al., 2018a). For any tensor function $g : \mathbb{R}^d \to (\mathbb{R}^d)^p$, we call it L Lipschitz continuous if for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$, $\|g(\mathbf{x}_1) - g(\mathbf{x}_2)\|_{\text{op}} \le L\|\mathbf{x} - \mathbf{y}\|_2$.

Next lemma is useful to control the difference between a tensor function and its Taylor series.

Lemma 3.2. *For any function $g : \mathbb{R}^d \to \mathbb{R}$, if $\nabla^p g$ exists and $\nabla^p g$ is L_p-Lipschitz continuous, then for any $\mathbf{y}, \mathbf{h} \in \mathbb{R}^d$, then for any $0 \le s \le p$, we have*

$$\left\|\nabla^s g(\mathbf{y}+\mathbf{h}) - \nabla^s g(\mathbf{y}) - \sum_{j=1}^{p-s} \frac{1}{j!}\langle \nabla^{s+j} g(\mathbf{y}), \mathbf{h}^{\otimes j}\rangle\right\|_{op} \le \frac{L_p}{(p-s+1)!}\|\mathbf{h}\|^{p-s+1}.$$

Proof. See Appendix A. □

Lemma 3.2 has many special forms. For instance, when we take $s = 0, p = 1$, we have the function smooth property used in first-order optimization. When we take $s = p = 2$, we have the Hessian smoothness property used in second-order optimization. Next we introduce our assumptions used in this work.

Assumption 3.3. For any $i \in [n]$, $\nabla^p f_i$ exists, and it is L_p-Lipschitz continuous.

Clearly, due to the triangle inequality, we conclude $\nabla^p f$ exists and is also L_p-Lipschitz continuous.

Assumption 3.4. Let the algorithm start at iteration $\mathbf{x}_0$, then we have $f(\mathbf{x}_0) - \inf_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) \le \Delta_f$.

Finally, given the access to up to p-th-order derivatives of the function f, we restate the high-order regularized method (HR) proposed by Birgin et al. (2017) here, with their convergence guarantee. Starting from $\mathbf{x}_0$, at iteration t, HR calculates the tuple of derivatives at iteration $\mathbf{x}_t$, which is $(\nabla f(\mathbf{x}_t), \dots, \nabla^p f(\mathbf{x}_t))$. Then HR updates $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \mathbf{h}_t$, where $\mathbf{h}_t$ can be set as global minimizer of the following p-th order regularized Taylor expansion of f at $\mathbf{x}_t$:

$$m_t(\mathbf{h}) := \sum_{j=1}^p \frac{1}{j!}\langle \nabla^j f(\mathbf{x}_t), \mathbf{h}^{\otimes j}\rangle + \frac{M}{(p+1)!}\|\mathbf{h}\|_2^{p+1},$$

where M is the regularization parameter. The following theorem suggests that HR only takes $O(\epsilon^{-(p+1)/p})$ number of iterations to find a stationary point.

Theorem 3.5 (Theorem 2.5, Birgin et al. 2017)**.** *By properly setting M, HR outputs $\mathbf{x}_T$ satisfying $\|\nabla f(\mathbf{x}_T)\|_2 \le \epsilon$, where $T = O(\Delta_f L_p^{1/p}/\epsilon^{(p+1)/p})$. Hence, HR finds ϵ-stationary points within $nT = O(nL_p^{1/p}/\epsilon^{(p+1)/p})$ number of IHO calls.*

4. Warm-up: Inexact Regularized p-th Order Optimization

Before proposing our main algorithms, we introduce a general inexact high-order optimization method in this section. We propose our algorithm as Algorithm 1. In general, Algorithm 1 runs an *inexact, constrained* high-order regularized method. At iteration t, Algorithm 1 computes estimators $\mathbf{J}_t^{(j)}, j = 1, \dots, p$ to approximate its derivatives $\nabla^j f(\mathbf{x}_t)$ at current iteration t. Then Algorithm 1 computes $\mathbf{h}_t$ as the minimizer of the regularized Taylor expansion m_t (4.1), within the ball $\|\mathbf{h}\|_2 \le r$, where r is the constraint radius. Algorithm 1 updates its iteration $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \mathbf{h}_t$. Now, Algorithm 1 either proceeds to next iteration or returns $\mathbf{x}_{t+1}$, if the norm of $\mathbf{h}_t$ is strictly less than the radius r. Note that similar constrained regularized approach has been applied for the $p = 2$ case in HVP-RVR (Arjevani et al., 2020). To better understand why Algorithm 1 needs such a ball constrain, and returns iterations based on $\|\mathbf{h}_t\|_2$, we propose the following two lemmas and their proofs.

Lemma 4.1. *Let t be the iteration where Algorithm 1 does not end. Then we have $\|\mathbf{h}_t\|_2 = r$ and*

$$f(\mathbf{x}_{t+1}) \le f(\mathbf{x}_t) - \frac{M}{2(p+1)!}r^{p+1} + \sum_{j=1}^p \frac{1}{j!}\|\nabla^j f(\mathbf{x}_t) - \mathbf{J}_t^{(j)}\|_{op} r^j.$$

Proof. First, since $\mathbf{h}_t$ is the minimizer of m_t over $\|\mathbf{h}\|_2 \le r$, we have $m_t(\mathbf{h}_t) = \inf_{\|\mathbf{h}\|_2 \le r} m_t(\mathbf{h}) \le m_t(\mathbf{0}) = 0$. Meanwhile, since Algorithm 1 does not end at t-th iteration, we have $\|\mathbf{h}_t\|_2 = r$. Then due to Lemma 3.2, we have

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \le \sum_{j=1}^p \frac{1}{j!}\langle \nabla^j f(\mathbf{x}_t), \mathbf{h}_t^{\otimes j}\rangle + \frac{L_p}{(p+1)!}\|\mathbf{h}_t\|_2^{p+1}$$

Algorithm 1 High-order Meta Algorithm

Require: Regularization parameter M, constraint radius r

for $t = 0, \dots, T-1$ **do**

Compute $\mathbf{J}_t^{(j)}, j = 1, \dots, p$ following Algorithm 2 or 3.

Let $\mathbf{h}_t$ be defined as follows:

$$\mathbf{h}_t \leftarrow \mathop{\mathrm{argmin}}_{\|\mathbf{h}\|_2 \le r} m_t(\mathbf{h}) := \sum_{j=1}^{p} \frac{1}{j!} \langle \mathbf{J}_t^{(j)}, \mathbf{h}^{\otimes j} \rangle + \frac{M}{(p+1)!} \|\mathbf{h}\|_2^{p+1}. \tag{4.1}$$

Set $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \mathbf{h}_t$

if $\|\mathbf{h}_t\|_2 < r$ **then return** $\mathbf{x}_{t+1}$

end for

$$= m_t(\mathbf{h}_t) + \sum_{j=1}^{p} \frac{1}{j!} \langle \nabla^j f(\mathbf{x}_t) - \mathbf{J}_t^{(j)}, \mathbf{h}_t^{\otimes j} \rangle - \frac{M - L_p}{(p+1)!} \|\mathbf{h}_t\|_2^{p+1}$$
$$\le -\frac{M}{2(p+1)!} \|\mathbf{h}_t\|_2^{p+1} + \sum_{j=1}^{p} \frac{1}{j!} \|\nabla^j f(\mathbf{x}_t) - \mathbf{J}_t^{(j)}\|_{\mathrm{op}} \|\mathbf{h}_t\|_2^j$$
$$= -\frac{M}{2(p+1)!} r^{p+1} + \sum_{j=1}^{p} \frac{1}{j!} \|\nabla^j f(\mathbf{x}_t) - \mathbf{J}_t^{(j)}\|_{\mathrm{op}} r^j,$$

where the second inequality holds due to the facts $m_t(\mathbf{h}_t) \le 0$ and $M \ge 2L_p$. That ends our proof. □

Lemma 4.2. *Suppose that Algorithm 1 ends at t-th iteration. Then we have*

$$\|\nabla f(\mathbf{x}_{t+1})\|_2 \le \frac{2M}{p!} r^p + \sum_{j=1}^{p} \frac{1}{(j-1)!} \|\nabla^j f(\mathbf{x}_t) - \mathbf{J}_t^{(j)}\|_{op} r^{j-1}.$$

Proof. Let $\mathcal{L}(\mathbf{h}, \lambda)$ be the Lagrangian function of the constrained minimization problem $\min m_t(\mathbf{h})$ with the constraint $\|\mathbf{h}\|_2 - r \le 0$, where λ is the dual variable. Then there exists a $\lambda_t > 0$, together with $\mathbf{h}_t$ satisfy the KKT condition, where

$$\lambda_t(\|\mathbf{h}_t\|_2 - r) = 0, \text{ (Complementary slackness)} \tag{4.2}$$
$$\mathbf{0} = \nabla m_t(\mathbf{h}_t) + \lambda_t \nabla(\|\mathbf{h}\|_2 - r)|_{\mathbf{h}=\mathbf{h}_t}, \text{ (Stationarity).} \tag{4.3}$$

Since Algorithm 1 ends at iteration t, then due to the design of the algorithm we have $\|\mathbf{h}\|_2 < r$. Then by (4.2), we have $\lambda_t = 0$. Substituting it to (4.3), we have $\mathbf{0} = \nabla m_t(\mathbf{h}_t)$, which leads to

$$\left\| \sum_{j=1}^{p} \frac{1}{(j-1)!} \langle \mathbf{J}_t^{(j)}, \mathbf{h}_t^{\otimes(j-1)} \rangle \right\|_2 = \frac{M}{p!} \|\mathbf{h}_t\|_2^p.$$

Furthermore, due to Lemma 3.2, we have

$$\left\| \nabla f(\mathbf{x}_{t+1}) - \sum_{j=1}^{p} \frac{1}{(j-1)!} \langle \nabla^j f(\mathbf{x}_t), \mathbf{h}_t^{\otimes(j-1)} \rangle \right\|_2 \le \frac{L_p}{p!} \|\mathbf{h}_t\|_2^p.$$

Then by triangle inequality, we have

$$\|\nabla f(\mathbf{x}_{t+1})\|_2$$
$$\le \left\| \nabla f(\mathbf{x}_{t+1}) - \sum_{j=1}^{p} \frac{1}{(j-1)!} \langle \nabla^j f(\mathbf{x}_t), \mathbf{h}_t^{\otimes(j-1)} \rangle \right\|_2 + \left\| \sum_{j=1}^{p} \frac{1}{(j-1)!} \langle \mathbf{J}_t^{(j)} - \nabla^j f(\mathbf{x}_t), \mathbf{h}_t^{\otimes(j-1)} \rangle \right\|_2 + \left\| \sum_{j=1}^{p} \frac{1}{(j-1)!} \langle \mathbf{J}_t^{(j)}, \mathbf{h}_t^{\otimes(j-1)} \rangle \right\|_2$$
$$\le \frac{M + L_p}{p!} \|\mathbf{h}_t\|_2^p + \sum_{j=1}^{p} \frac{1}{(j-1)!} \|\nabla^j f(\mathbf{x}_t) - \mathbf{J}_t^{(j)}\|_{\mathrm{op}} \|\mathbf{h}_t\|_2^{j-1}$$
$$\le \frac{2M}{p!} r^p + \sum_{j=1}^{p} \frac{1}{(j-1)!} \|\nabla^j f(\mathbf{x}_t) - \mathbf{J}_t^{(j)}\|_{\mathrm{op}} r^{j-1},$$

where the last inequality holds since $\|\mathbf{h}_t\|_2 \le r$ and $M \ge L_p$. □

Lemmas 4.1 and 4.2 show the power of applying a ball constraint over $\mathbf{h}_t$ to HR. Assume that the derivative estimation error $\mathbf{J}_t^{(j)} - \nabla^j f(\mathbf{x}_t)$ is small enough. Lemma 4.1 suggests that the function value $f(\mathbf{x})$ will decrease at least $O(Mr^{p+1})$ per step. On the other hand, Lemma 4.2 suggests that the norm of the gradient of the final output x_{T+1} is of order $O(Mr^p)$. Thus, Algorithm 1 will end within $O(\Delta_f/(Mr^{p+1}))$ number of iterations to find an $O(Mr^p)$-stationary point. By setting $Mr^p \sim \epsilon$ would guarantee the finding of ϵ-stationary points. Without such a constrain, we can not guarantee that the final output iteration is an ϵ-stationary point.

Difference between Algorithm 1 and Cartis et al. (2020a) Cartis et al. (2020b) Cartis et al. (2017) Here we highlight the difference between our Algorithm 1 and several related works. Compared with Cartis et al. (2017), our Algorithm 1 introduces an additional constraint $\|\mathbf{h}\|_2 \le r$, which guarantees that the approximation error of our gradient estimator can be well-controlled. Compared with Cartis et al. (2020a), Algorithm 1 allows the inexact derivative estimators. Compared with Cartis et al. (2020b), our algorithm uses a regularized version of the high-order Taylor expansion of f.

Implementation of Algorithm 1 for $p = 1, 2$ We discuss how to solve (4.1) for $p = 1, 2$ cases, and we leave to solve the general $p > 2$ cases as future work. For the $p = 1$ case, (4.1) becomes a standard stochastic gradient descent step which can be computed in $O(d)$ time. For the $p = 2$ case, we solve (4.1) by the method of Lagrange multipliers. By the proof of Lemma 4.2, we know that $\mathbf{h}_t$ satisfies $\|\mathbf{h}_t\|_2 \le r$, (4.2) and (4.3). Therefore, if $\|\mathbf{h}_t\|_2 = r$, then by (4.3) we have

$$\mathbf{J}_t^1 + \mathbf{J}_t^2\mathbf{h}_t + \frac{M}{2}\|\mathbf{h}_t\|_2\mathbf{h}_t + \lambda_t\frac{\mathbf{h}_t}{\|\mathbf{h}_t\|} = \mathbf{0}$$
$$\Rightarrow \mathbf{h}_t = u_t(\lambda_t) := -(\mathbf{J}_t^2 + Mr/2\mathbf{I} + \lambda_t/r\mathbf{I})^\dagger\mathbf{J}_t^1,$$

where $\mathbf{A}^\dagger$ denotes the pseudoinverse of $\mathbf{A}$. Since $\|\mathbf{h}_t\|_2 = r$, we have $\|u_t(\lambda_t)\|_2 = r$. Therefore, based on whether the equation $\|u_t(\lambda)\|_2 = r$ has a nonnegative solution, (4.1) can be solved as follows:

- If $\|u_t(\lambda)\|_2 = r$ has a positive solution $\lambda^* > 0$, we set $\lambda_t = \lambda^*$ and $\mathbf{h}_t = u_t(\lambda_t)$. Such a calculation can be done by computing the product between a matrix inverse and a vector with $O(d^2)$ complexity.
- If $\|u_t(\lambda)\|_2 = r$ does not have a positive solution, we must have $\|\mathbf{h}_t\|_2 < r$, then (4.1) becomes the standard cubic regularization subproblem, which can be solved with $O(d^3)$ complexity (Nesterov & Polyak, 2006).

5. Stochastic High-order Derivative Estimator

In Section 4 we have proposed Algorithm 1 as a general framework which can be applied with any inexact derivative estimators. In this section we propose two algorithms that provide different construction of the derivative estimators.

5.1. One-point Taylor Expansion Estimator

We propose our first algorithm in Algorithm 2. Algorithm 2 adapts the two-layer-loop framework which has been widely used in variance reduction-based algorithms (Johnson & Zhang, 2013). Specifically, Algorithm 2 maintains a *reference point* denoted by $\widetilde{\mathbf{x}}$, which is repeatedly updated as the current iteration $\mathbf{x}_t$ with frequency m. The following lemma suggests at any iteration, the distance between the current iteration and the last reference point will be upper bounded by both the frequency and the constraint radius.

Algorithm 2 OP-TE

Require: Reference point update frequency m, batch size B.

1: **if** $t = 0 \mod m$ **then**
2: $\quad\mathbf{J}_t^{(j)} \leftarrow \nabla^j f(\mathbf{x}_t), \forall 1 \le j \le p,\ \widetilde{\mathbf{x}} \leftarrow \mathbf{x}_t$
3: **else**
4: $\quad$Sample $\mathcal{S}_t \in [n]$, $|\mathcal{S}_t| = B$
5: $\quad$We set $\mathbf{J}_t^{(1)}$ as

$$\mathbf{J}_t^{(1)} \leftarrow \nabla f_{\mathcal{S}_t}(\mathbf{x}_t) - \nabla f_{\mathcal{S}_t}(\widetilde{\mathbf{x}}) + \nabla f(\widetilde{\mathbf{x}}) + \sum_{s=1}^{p-1}\frac{1}{s!}\langle\nabla^{s+1}f(\widetilde{\mathbf{x}}) - \nabla^{s+1}f_{\mathcal{S}_t}(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle.$$

$\quad$For $2 \le j \le p$, we set

$$\mathbf{J}_t^{(j)} \leftarrow \nabla^j f(\widetilde{\mathbf{x}}) + \sum_{s=1}^{p-j}\frac{1}{s!}\langle\nabla^{j+s}f(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle.$$

6: **end if**

Lemma 5.1. *For any t, let $\widetilde{\mathbf{x}}$ be the reference point at t-th iteration, then we have $\|\widetilde{\mathbf{x}} - \mathbf{x}_t\|_2 \le mr$.*

Proof. By the definition of $\widetilde{\mathbf{x}}$ we know that there exists an q such that $\widetilde{\mathbf{x}} = \mathbf{x}_q$ and $q \le t \le q + m$. Using the fact that $\widetilde{\mathbf{x}} - \mathbf{x}_t = \sum_{i=q}^{t-1}(\mathbf{x}_i - \mathbf{x}_{i+1}) = \sum_{i=q}^{t-1}\mathbf{h}_i$, then we have

$$\|\widetilde{\mathbf{x}} - \mathbf{x}_t\|_2 \le (t-q)r \le mr,$$

where the first inequality holds due to triangle inequality, the second one holds since $\|\mathbf{h}_i\|_2 \le r$ and the last one holds since $0 \le t - q \le m$. $\square$

When the reference point is updated, Algorithm 2 computes and stores derivatives of function f from order 1 to p by calling IHO n times. After that, at each iteration, Algorithm 2 samples an index subset $\mathcal{S}_t$ with cardinality $|\mathcal{S}_t| = B$. Then Algorithm 2 sets the derivative estimator $\mathbf{J}_t^{(j)}$ for $j = 1$ and $j \ge 2$ separately. **Case I:** $j \ge 2$**.** Algorithm 2 constructs $\mathbf{J}_t^{(j)}$ as the $(p-j)$*-th order Taylor series* of derivative function $\nabla^j f$ at point $\widetilde{\mathbf{x}}$. There are a few key points needs to be noticed for our construction. First, such estimators *do not* use any information of the derivatives at current iteration $\mathbf{x}_t$, since it only uses the stored derivatives $\nabla^{j+s}f(\widetilde{\mathbf{x}})$ at $\widetilde{\mathbf{x}}$. Second, such estimators are *deterministic but biased*, since it does not involve any randomness. In a sharp contrast, existing estimators such as stochastic Hessian in SVRC (Zhou et al., 2018a) or STR (Shen et al., 2019) for the setting $p = 2$, use information at current iteration to build their stochastic estimators. It seems that our estimators may be inferior than previous estimators. However, the following lemma characterizes the estimation error of our estimators and suggests that our estimators have been "accurate enough".

Lemma 5.2. *For $2 \le j \le p$ and any t, we have*

$$\|\mathbf{J}_t^{(j)} - \nabla^j f(\mathbf{x}_t)\|_{op} \le \frac{L_p m^{p-j+1} r^{p-j+1}}{(p-j+1)!}.$$

Proof. By Lemma 3.2, we have

$$\begin{aligned}
&\|\mathbf{J}_t^{(j)} - \nabla^j f(\mathbf{x}_t)\|_{\text{op}} \\
&= \left\|\nabla^j f(\widetilde{\mathbf{x}}) + \sum_{s=1}^{p-j} \frac{1}{s!}\langle \nabla^{j+s} f(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle - \nabla^j f(\mathbf{x}_t)\right\|_{\text{op}} \\
&\le \frac{L_p}{(p-j+1)!}\|\widetilde{\mathbf{x}} - \mathbf{x}_t\|_2^{p-j+1} \\
&\le \frac{L_p m^{p-j+1} r^{p-j+1}}{(p-j+1)!}, \qquad (5.1)
\end{aligned}$$

where the last inequality holds due to Lemma 5.1. □

Existing high-order estimators such as SVRC are stochastic. To prove the estimation error, existing works need the concentration inequality for matrices (Chen et al., 2012; Tropp, 2016; Zhou et al., 2018a) or tensors (Vershynin, 2020), which depend on the dimension d. Such a dependence is even unavoidable for the case where the stochastic matrices are Gaussian series (Sec 4.1.2, Tropp et al. 2015). To contrast with, our estimators enjoy a *dimension-free* error bound, which leads to a dimension-free complexity result for our algorithm in the later analysis.

Case II: $j = 1$**.** We now come to the gradient estimator case where $j = 1$. Algorithm 2 constructs its gradient estimator in a different way. Intuitively speaking, Algorithm 2 estimates $\nabla f(\mathbf{x}_t)$ by firstly taking the following difference decomposition:

$$\begin{aligned}
\nabla f(\mathbf{x}_t) &= \nabla f_{\mathcal{S}_t}(\mathbf{x}_t) + \nabla f(\widetilde{\mathbf{x}}) - \nabla f_{\mathcal{S}_t}(\widetilde{\mathbf{x}}) \\
&\quad - \underbrace{(\nabla f_{\mathcal{S}_t}(\mathbf{x}_t) - \nabla f_{\mathcal{S}_t}(\widetilde{\mathbf{x}}))}_{I_1} + \underbrace{\nabla f(\mathbf{x}_t) - \nabla f(\widetilde{\mathbf{x}})}_{I_2},
\end{aligned}$$

then Algorithm 2 estimates I_1 with the Taylor series of $\nabla f_{\mathcal{S}_t}(\mathbf{x})$ expanded at $\widetilde{\mathbf{x}}$, I_2 with the Taylor series of $\nabla f(\mathbf{x})$ expanded at $\widetilde{\mathbf{x}}$, that is,

$$\begin{aligned}
I_1 &\approx \sum_{s=1}^{p-j} \frac{1}{s!}\langle \nabla^{j+s} f_{\mathcal{S}_t}(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle \\
I_2 &\approx \sum_{s=1}^{p-j} \frac{1}{s!}\langle \nabla^{j+s} f(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle.
\end{aligned}$$

Algorithm 3 TP-TE

Require: Reference point update frequency m,

1: **if** $t = 0 \mod m$ **then**

2: $\quad \mathbf{J}_t^{(j)} \leftarrow \nabla^j f(\mathbf{x}_t), \forall 1 \le j \le p,\ \widetilde{\mathbf{x}} \leftarrow \mathbf{x}_t$

3: **else**

4: $\quad$ Sample $\mathcal{S}_t \in [n], |\mathcal{S}_t| = B$

5: $\quad$ We set $\mathbf{J}_t^{(j)}$ as

$$\begin{aligned}
\mathbf{J}_t^{(1)} &\leftarrow \mathbf{J}_{t-1}^{(1)} + \nabla f_{\mathcal{S}_t}(\mathbf{x}_t) - \nabla f_{\mathcal{S}_t}(\mathbf{x}_{t-1}) \\
&\quad + \sum_{s=1}^{p-1} \frac{1}{s!}\langle \mathbf{H}_t^{(s)}, (\mathbf{x}_t - \mathbf{x}_{t-1})^{\otimes s}\rangle, \\
\mathbf{H}_t^{(s)} &= \sum_{i=0}^{p-1-s} \frac{1}{i!}\langle \nabla^{s+i+1} f(\widetilde{\mathbf{x}}) - \nabla^{s+i+1} f_{\mathcal{S}_t}(\widetilde{\mathbf{x}}), \\
&\quad (\mathbf{x}_{t-1} - \widetilde{\mathbf{x}})^{\otimes i}\rangle.
\end{aligned}$$

$\quad$ For $2 \le j \le p$, we set

$$\mathbf{J}_t^{(j)} \leftarrow \nabla^j f(\widetilde{\mathbf{x}}) + \sum_{s=1}^{p-j} \frac{1}{s!}\langle \nabla^{j+s} f(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle.$$

6: **end if**

Such a construction strategy has also appeared in SVRG (Johnson & Zhang, 2013) for the setting $p = 1$ or semi-stochastic gradient in SVRC for the setting $p = 2$. The following lemma suggests that, due to the use of stochastic batch $\mathcal{S}_t$, the estimator error of the $\mathbf{J}_t^{(1)}$ is smaller by a factor of $\sqrt{B}$, compared with the previous high-order estimators.

Lemma 5.3. *Let $\mathbf{J}_t^{(1)}$ be the estimate generated by Algorithm 2. Then with probability at least $1 - T\delta$, we have*

$$\|\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t)\|_2 \le 4\sqrt{\log(4/\delta)}\frac{L_p m^p r^p}{\sqrt{B}p!}.$$

Based on Lemma 5.2 and 5.3, we propose our complexity analysis for Algorithm 1 equipeed with estimators from Algorithm 2.

Theorem 5.4. *There exist functions $g_i, i = 1\dots 4$ that only depend on p such that, by setting m, B, M, r as follows:*

$$\begin{aligned}
m &= \frac{g_1(p)n^{1/3}}{\log^{1/3}(4/\delta)},\ B = g_2(p)\log^{2/3}(4/\delta)n^{2/3}, \\
M &= \frac{g_3(p)n^{\frac{p-1}{3}}L_p}{\log^{\frac{p-1}{3}}(4/\delta)},\ r = \frac{g_4(p)\log^{(p-1)/(3p)}(4/\delta)\epsilon^{1/p}}{L_p^{1/p}n^{(p-1)/(3p)}},
\end{aligned}$$

and set

$$T = \Delta_f / \left(\frac{M}{4(p+1)!}r^{p+1}\right),$$

then with probability at least $1 - T\delta$, Algorithm 1 with Algorithm 2 ends in T steps and outputs an ϵ-stationary point. Meanwhile, there exists a function g that only depends on p such that the number of total IHO calls is bounded as

$$g(p) \cdot \log^{(p+1)/(3p)}(4/\delta) \cdot \frac{\Delta_f L_p^{1/p} n^{(3p-1)/(3p)}}{\epsilon^{(p+1)/p}}.$$

Remark 5.5. Regarding p, δ, Δ_f, L_p as constants, Theorem 5.4 suggests an $O(n^{(3p-1)/(3p)}/\epsilon^{(p+1)/p})$ *dimension-free* oracle complexity for Algorithm 2. Compared with HR (Birgin et al., 2017), our result matches its dependence on ϵ, which is optimal (Carmon et al., 2017). Meanwhile, our result improves HR's dependence on n by a factor of $n^{1/(3p)}$.

Remark 5.6. When $p = 1$, our algorithm reduces to SVRG with an $O(n^{2/3}/\epsilon^2)$ gradient complexity, which matches results in Allen-Zhu & Hazan (2016); Reddi et al. (2016).

5.2. Two-point Taylor Expansion Estimator

We propose our second algorithm in Algorithm 3. Algorithm 3 also adapts a two-layer-loop framework as Algorithm 2, and maintains a reference point $\widetilde{\mathbf{x}}$ with a m-step update frequency. Algorithm 3 only computes the full derivative $\nabla^j f$ when updating the reference points. At other iterations, it utilizes function information $f_i, i \in \mathcal{S}_t$ where $|\mathcal{S}_t| = B$ is a subset of $[n]$. Algorithm 3 constructs the high-order derivative estimators the same as Algorithm 2, and constructs its gradient estimator in a different way. The intuition of its gradient estimator is demonstrated in the following equation:

$$\begin{aligned}
&\nabla f(\mathbf{x}_t) \\
&= \nabla f(\mathbf{x}_{t-1}) + \nabla f_{\mathcal{S}_t}(\mathbf{x}_t) - \nabla f_{\mathcal{S}_t}(\mathbf{x}_{t-1}) \\
&- (\nabla f_{\mathcal{S}_t}(\mathbf{x}_t) - \nabla f_{\mathcal{S}_t}(\mathbf{x}_{t-1})) + \nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}).
\end{aligned}$$

Based on above decomposition, Algorithm 3 constructs the gradient estimator by replacing $\nabla f(\mathbf{x}_t)$ with $\mathbf{J}_t^{(1)}$ and $\nabla f(\mathbf{x}_{t-1})$ with $\mathbf{J}_{t-1}^{(1)}$, which gives us the following update rule:

$$\begin{aligned}
&\mathbf{J}_t^{(1)} \\
&\leftarrow \mathbf{J}_{t-1}^{(1)} + \nabla f_{\mathcal{S}_t}(\mathbf{x}_t) - \nabla f_{\mathcal{S}_t}(\mathbf{x}_{t-1}) \\
&- \underbrace{(\nabla f_{\mathcal{S}_t}(\mathbf{x}_t) - \nabla f_{\mathcal{S}_t}(\mathbf{x}_{t-1}))}_{I_1} + \underbrace{\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1})}_{I_2}.
\end{aligned}$$

To estimate I_1 and I_2, a natural way is to replace them with the Taylor series of $\nabla f_{\mathcal{S}_t}(\mathbf{x})$ and $\nabla f(\mathbf{x})$ at point $\mathbf{x}_{t-1}$. Since $\mathbf{x}_{t-1}$ is 'closer' to the current iteration $\mathbf{x}_t$ compared with $\widetilde{\mathbf{x}}$, thus intuitively speaking, it should provide a preciser estimation than Algorithm 2. However, we can not directly use above Taylor series since they require to compute the full derivatives $\nabla^j f(\mathbf{x}_{t-1})$. To work around this issue, instead of using $\nabla^j f(\mathbf{x}_{t-1})$, Algorithm 3 computes its $(p-j)$-th order Taylor series expanded at $\widetilde{\mathbf{x}}$ as a replacement, that is,

$$\nabla^j f(\mathbf{x}_{t-1}) \approx \nabla^j f(\widetilde{\mathbf{x}}) + \sum_{i=1}^{p-j} \frac{1}{i!} \langle \nabla^{j+i} f(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes i} \rangle.$$

Next lemma provides the estimation error of $\mathbf{J}_t^{(1)}$.

Lemma 5.7. *Let $\mathbf{J}_t^{(1)}$ be the estimate generated by Algorithm 3. Then with probability at least $1 - T\delta$, for all $1 \leq t \leq T$, we have*

$$\|\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t)\|_2 \leq 6\sqrt{\log(4/\delta)} \frac{2^p m^{p-1/2} L_p r^p}{p!\sqrt{B}}.$$

Compared with Lemma 5.3, the error bound in Lemma 5.7 is improved by a factor of $\sqrt{m}$. Next we propose the theorem for Algorithm 3.

Theorem 5.8. *There exist functions $g_i, i = 1 \ldots 2$ that only depend on p such that, by setting m, B, M, r as follows: let $m \geq p\sqrt{n}$ and*

$$\begin{aligned}
&B = n/m, \ M = g_1(p) \frac{\sqrt{\log(4/\delta)} L_p m^p}{\sqrt{n}}, \\
&r = g_2(p) \frac{n^{1/(2p)} \epsilon^{1/p}}{\log^{1/(2p)}(4/\delta) m L_p^{1/p}},
\end{aligned}$$

and set

$$T = \Delta_f \Big/ \left(\frac{M}{4(p+1)!} r^{p+1} \right),$$

then with probability at least $1 - T\delta$, Algorithm 1 with Algorithm 2 ends in T steps and outputs an ϵ-stationary point. Meanwhile, there exists a function g that only depends on p such that the number of total IHO calls is bounded as

$$g(p) \cdot \log^{1/(2p)}(4/\delta) \cdot \frac{\Delta_f L_p^{1/p} n^{(2p-1)/(2p)}}{\epsilon^{(p+1)/p}}.$$

Remark 5.9. Regarding p, δ, Δ_f, L_p as constants, Theorem 5.8 suggests an $O(n^{(2p-1)/(2p)}/\epsilon^{(p+1)/p})$ oracle complexity for Algorithm 3. Compared with Algorithm 2, the complexity of Algorithm 3 also has an optimal dependence on ϵ. Meanwhile, it improves the dependence on n by a factor of $n^{1/(6p)}$.

Remark 5.10. When $p = 1$, Algorithm 2 reduces to SPIDER with an $O(n^{1/2}/\epsilon^2)$ gradient complexity, which matches the result in Fang et al. (2018). When $p = 2$, the gradient estimator reduces to that of STR with an $O(n^{3/4}/\epsilon^{3/2})$ second-order oracle complexity, and the overall complexity matches that in Shen et al. (2019).

Remark 5.11. The IHO complexity of our Algorithm 3 is *dimension-free*, unlike previous result which depends on d

logarithmically ($p = 2$, Shen et al. 2019). Meanwhile, note that Algorithm 3 allows a free selection of m. By selecting $m = n$, the batch size B is set to 1, which further suggests that Algorithm 3 supports the use of a *small batch size*. This is the first result in stochastic high-order optimization that does not need to access a large batch size, unlike all previous works.

Optimal oracle complexity? We briefly discuss the optimality of Theorem 5.8. Emmenegger et al. (2021) proposed an $\Omega(n^{(p-1)/(2p)}/\epsilon^{(p+1)/p})$ lower bound of IHO complexity for any algorithm finding ϵ-stationary points, and there exists a $\sqrt{n}$ gap between this lower bound and our upper bound. We believe such a gap is caused by our Assumption 3.3 and can be potentially fixed by considering a proper function class. For instance, when $p = 1$, our upper bound keeps unchanged under a slightly larger function class called *average-smooth function class* (Fang et al., 2018; Zhou et al., 2018c), where Assumption 3.3 is replaced by the average-smoothness assumption, $\mathbb{E}_i\|\nabla f_i(\mathbf{x})-\nabla f_i(\mathbf{y})\|_2^2 \leq L_2^2\|\mathbf{x}-\mathbf{y}\|_2^2$. Meanwhile, the lower bound can be substantially improved to $\Omega(n^{1/2}/\epsilon^2)$ that matches our upper bound. We leave to fix such a gap as future work.

6. Conclusion

In this work, we study the stochastic high-order methods for finite-sum nonconvex optimization problems. We propose two algorithms OP-TE and TP-TE, where TP-TE finds ϵ-stationary points within $O(n^{(2p-1)/(2p)}/\epsilon^{(p+1)/p})$ oracle complexity. Our algorithm is the first one that strictly improves the deterministic high-order optimization algorithm, and it is also the first algorithm that enjoys a dimension-free oracle complexity without using a large batch size.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. DZ and QG are partially supported by the National Science Foundation CAREER Award 1906169. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

Agarwal, A. and Bottou, L. A lower bound for the optimization of finite sums. In *International conference on machine learning*, pp. 78–86. PMLR, 2015.

Allen-Zhu, Z. and Hazan, E. Variance reduction for faster non-convex optimization. In *International Conference on Machine Learning*, pp. 699–707, 2016.

Arjevani, Y., Carmon, Y., Duchi, J. C., Foster, D. J., Sekhari, A., and Sridharan, K. Second-order information in nonconvex stochastic optimization: Power and limitations. In *Conference on Learning Theory*, pp. 242–299. PMLR, 2020.

Bellavia, S., Gurioli, G., Morini, B., and Toint, P. L. Trust-region algorithms: probabilistic complexity and intrinsic noise with applications to subsampling techniques. *arXiv preprint arXiv:2112.06176*, 2021.

Birgin, E. G., Gardenghi, J., Martínez, J. M., Santos, S. A., and Toint, P. L. Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models. *Mathematical Programming*, 163 (1-2):359–368, 2017.

Birgin, E. G., Gardenghi, J., Martínez, J. M., and Santos, S. A. On the use of third-order models with fourth-order regularization for unconstrained optimization. *Optimization Letters*, 14(4):815–838, 2020.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Lower bounds for finding stationary points ii: First-order methods. *arXiv preprint arXiv:1711.00841*, 2017.

Cartis, C., Gould, N., and Toint, P. L. Second-order optimality and beyond: characterization and evaluation complexity in nonconvex convexly-constrained optimization. *Preprint RAL-P-2016-008, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England*, 2016.

Cartis, C., Gould, N. I., and Toint, P. L. Improved second-order evaluation complexity for unconstrained nonlinear optimization using high-order regularized models. *arXiv preprint arXiv:1708.04044*, 2017.

Cartis, C., Gould, N. I., and Toint, P. L. Sharp worst-case evaluation complexity bounds for arbitrary-order nonconvex optimization with inexpensive constraints. *SIAM Journal on Optimization*, 30(1):513–541, 2020a.

Cartis, C., Gould, N. I. M., and Toint, P. L. Strong evaluation complexity of an inexact trust-region algorithm for arbitrary-order unconstrained nonconvex optimization. *arXiv preprint arXiv:2011.00854*, 2020b.

Chen, R. Y., Gittens, A., and Tropp, J. A. The masked sample covariance estimator: an analysis using matrix concentration inequalities. *Information and Inference: A Journal of the IMA*, 1(1):2–20, 2012.

Cutkosky, A. and Orabona, F. Momentum-based variance reduction in non-convex sgd. *arXiv preprint arXiv:1905.10018*, 2019.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in*

Neural Information Processing Systems, pp. 1646–1654, 2014.

Emmenegger, N., Kyng, R., and Zehmakan, A. N. On the oracle complexity of higher-order smooth non-convex finite-sum optimization. *arXiv preprint arXiv:2103.05138*, 2021.

Fang, C., Li, C. J., Lin, Z., and Zhang, T. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pp. 687–697, 2018.

Hillar, C. J. and Lim, L.-H. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):45, 2013.

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

Kohler, J. M. and Lucchi, A. Sub-sampled cubic regularization for non-convex optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1895–1904. PMLR, 2017.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Nesterov, Y. and Polyak, B. T. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pp. 2613–2621, 2017.

Nguyen, L. M., van Dijk, M., Phan, D. T., Nguyen, P. H., Weng, T.-W., and Kalagnanam, J. R. Optimal finite-sum smooth non-convex optimization with sarah. *arXiv preprint arXiv:1901.07648*, 2019.

Pinelis, I. Optimum bounds for the distributions of martingales in banach spaces. *The Annals of Probability*, pp. 1679–1706, 1994.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. Stochastic variance reduction for nonconvex optimization. In *International Conference on Machine Learning*, pp. 314–323, 2016.

Roux, N. L., Schmidt, M., and Bach, F. R. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pp. 2663–2671, 2012.

Shen, Z., Zhou, P., Fang, C., and Ribeiro, A. A stochastic trust region method for non-convex minimization. *arXiv preprint arXiv:1903.01540*, 2019.

Tropp, J. A. The expected norm of a sum of independent random matrices: An elementary approach. In *High Dimensional Probability VII*, pp. 173–202. Springer, 2016.

Tropp, J. A. et al. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015.

Vershynin, R. Concentration inequalities for random tensors. *Bernoulli*, 26(4):3139–3162, 2020.

Wang, Z., Ji, K., Zhou, Y., Liang, Y., and Tarokh, V. Spiderboost: A class of faster variance-reduced algorithms for nonconvex optimization. *arXiv preprint arXiv:1810.10690*, 2018a.

Wang, Z., Zhou, Y., Liang, Y., and Lan, G. Sample complexity of stochastic variance-reduced cubic regularization for nonconvex optimization. *arXiv preprint arXiv:1802.07372*, 2018b.

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

Xu, P., Roosta-Khorasani, F., and Mahoney, M. W. Newton-type methods for non-convex optimization under inexact hessian information. *arXiv preprint arXiv:1708.07164*, 2017.

Zhang, J., Xiao, L., and Zhang, S. Adaptive stochastic variance reduction for subsampled newton method with cubic regularization. *arXiv preprint arXiv:1811.11637*, 2018.

Zhang, X., Ling, C., and Qi, L. The best rank-1 approximation of a symmetric tensor and related spherical optimization problems. *SIAM Journal on Matrix Analysis and Applications*, 33(3):806–821, 2012.

Zhou, D. and Gu, Q. Lower bounds for smooth nonconvex finite-sum optimization. In *International Conference on Machine Learning*, pp. 7574–7583. PMLR, 2019.

Zhou, D., Xu, P., and Gu, Q. Stochastic variance-reduced cubic regularized newton methods. In *International Conference on Machine Learning*, pp. 5990–5999. PMLR, 2018a.

Zhou, D., Xu, P., and Gu, Q. Sample efficient stochastic variance-reduced cubic regularization method. *arXiv preprint arXiv:1811.11989*, 2018b.

Zhou, D., Xu, P., and Gu, Q. Stochastic nested variance reduced gradient descent for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pp. 3922–3933, 2018c.

A. Proof of Lemma 3.2

Suppose the claim holds for $s+1$. For s, let $h(\lambda) := \nabla^s f(\mathbf{y}+\lambda\mathbf{h})$. Then we have

$$\begin{aligned}
&\nabla^s f(\mathbf{y}+\mathbf{h}) - \nabla^s f(\mathbf{y}) - \sum_{j=1}^{p-s}\frac{1}{j!}\langle \nabla^{s+j} f(\mathbf{y}), \mathbf{h}^{\otimes j}\rangle \\
&= \int_0^1 \langle \nabla^{s+1} f(\mathbf{y}+\lambda\mathbf{h}), \mathbf{h}\rangle d\lambda - \langle \nabla^{s+1} f(\mathbf{y}), \mathbf{h}\rangle - \sum_{j=1}^{p-s-1}\frac{1}{(j+1)!}\langle \nabla^{s+1+j} f(\mathbf{y}), \mathbf{h}^{\otimes (j+1)}\rangle \\
&= \left\langle \mathbf{h}, \int_0^1 \nabla^{s+1} f(\mathbf{y}+\lambda\mathbf{h}) d\lambda - \nabla^{s+1} f(\mathbf{y}) - \sum_{j=1}^{p-s-1}\frac{1}{(j+1)!}\langle \nabla^{s+1+j} f(\mathbf{y}), \mathbf{h}^{\otimes j}\rangle \right\rangle \\
&= \left\langle \mathbf{h}, \int_0^1 \left(\nabla^{s+1} f(\mathbf{y}+\lambda\mathbf{h}) - \nabla^{s+1} f(\mathbf{y}) - \sum_{j=1}^{p-s-1}\frac{1}{j!}\langle \nabla^{s+1+j} f(\mathbf{y}), (\lambda\mathbf{h})^{\otimes j}\rangle \right) d\lambda \right\rangle,
\end{aligned}$$

where the last line we use the fact that $\int_0^1 \lambda^j = 1/(j+1)$. Therefore, by applying Cauchy-Schwarz inequality we have

$$\begin{aligned}
&\left\| \nabla^s f(\mathbf{y}+\mathbf{h}) - \nabla^s f(\mathbf{y}) - \sum_{j=1}^{p-s}\frac{1}{j!}\langle \nabla^{s+j} f(\mathbf{y}), \mathbf{h}^{\otimes j}\rangle \right\|_{\text{op}} \\
&\le \|\mathbf{h}\|_2 \int_0^1 \left\| \nabla^{s+1} f(\mathbf{y}+\lambda\mathbf{h}) - \nabla^{s+1} f(\mathbf{y}) - \sum_{j=1}^{p-s-1}\frac{1}{j!}\langle \nabla^{s+1+j} f(\mathbf{y}), (\lambda\mathbf{h})^{\otimes j}\rangle \right) \right\|_{\text{op}} d\lambda \\
&\le \|\mathbf{h}\|_2 \int_0^1 \frac{L_p \lambda^{p-s}}{(p-s)!}\|\mathbf{h}\|^{p-s} d\lambda \\
&= \frac{L_p}{(p-s+1)!}\|\mathbf{h}\|^{p-s+1}.
\end{aligned}$$

That ends our proof by applying the induction from $s = p$ to 0.

B. Proof of results in Section 5

In this section we prove lemmas and theorems in Section 5. We need the following lemma.

Lemma B.1 (Theorem 3.5, Pinelis 1994)**.** *Let $\boldsymbol{\epsilon}_{1:k} \in \mathbb{R}^d$ be a vector-valued martingale difference sequence with respect to $\mathcal{F}_k$, i.e., for each $k \in [K]$, $\mathbb{E}[\boldsymbol{\epsilon}_k|\mathcal{F}_k] = 0$ and $\|\boldsymbol{\epsilon}_k\|_2 \le B_k$, then we have given $\delta \in (0,1)$, w.p. $1-\delta$,*

$$\left\| \sum_{i=1}^K \boldsymbol{\epsilon}_k \right\|_2^2 \le 4\log(4/\delta)\sum_{i=1}^K B_k^2.$$

B.1. Proof of Lemma 5.3

Proof of Lemma 5.3. We bound the difference between $\mathbf{J}_t^{(1)}$ and $\nabla f(\mathbf{x}_t)$. First, we decompose $\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t)$ as follows:

$$\begin{aligned}
&\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t) \\
&= \nabla f_{\mathcal{S}_t}(\mathbf{x}_t) - \nabla f_{\mathcal{S}_t}(\widetilde{\mathbf{x}}) + \nabla f(\widetilde{\mathbf{x}}) - \nabla f(\mathbf{x}_t) + \sum_{s=1}^{p-1}\frac{1}{s!}\langle \nabla^{s+1} f(\widetilde{\mathbf{x}}) - \nabla^{s+1} f_{\mathcal{S}_t}(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle \\
&= \frac{1}{|\mathcal{S}_t|}\sum_{i\in\mathcal{S}_t}\Bigg[\underbrace{\nabla f_i(\mathbf{x}_t) - \nabla f_i(\widetilde{\mathbf{x}}) + \nabla f(\widetilde{\mathbf{x}}) - \nabla f(\mathbf{x}_t) + \sum_{s=1}^{p-1}\frac{1}{s!}\langle \nabla^{s+1} f(\widetilde{\mathbf{x}}) - \nabla^{s+1} f_i(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle}_{I_i}\Bigg]. \qquad \text{(B.1)}
\end{aligned}$$

On the one hand, since $\mathcal{S}_t$ is sampled from $[n]$ i.i.d., then we have $\mathbb{E}_i[I_i] = \mathbf{0}$. On the other hand, I_i has the following upper bound:

$$\begin{aligned}
\|I_i\|_2 &\le \bigg\|\nabla f_i(\mathbf{x}_t) - \nabla f_i(\widetilde{\mathbf{x}}) - \sum_{s=1}^{p-1}\frac{1}{s!}\langle \nabla^{s+1} f_i(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle\bigg\|_2 \\
&\quad + \bigg\|\nabla f(\mathbf{x}_t) - \nabla f(\widetilde{\mathbf{x}}) - \sum_{s=1}^{p-1}\frac{1}{s!}\langle \nabla^{s+1} f(\widetilde{\mathbf{x}}), (\mathbf{x}_t - \widetilde{\mathbf{x}})^{\otimes s}\rangle\bigg\|_2 \\
&\le \frac{2L_p}{p!}\|\mathbf{x}_t - \widetilde{\mathbf{x}}\|_2^p \\
&\le \frac{2L_p m^p r^p}{p!},
\end{aligned} \tag{B.2}$$

where the first inequality holds due to triangle inequality, the second one holds due to Lemma 3.2 and the last one holds due to Lemma 5.1. Therefore, by Lemma B.1, for each t, with probability at least $1-\delta$,

$$\|\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t)\|_2 \le \frac{1}{B}\cdot 4\sqrt{\log(4/\delta)}\frac{L_p m^p r^p}{p!}\cdot\sqrt{B} = 4\sqrt{\log(4/\delta)}\frac{L_p m^p r^p}{\sqrt{B}p!}.$$

Taking union bound over T steps ends our proof. $\square$

B.2. Proof of Lemma 5.7

Proof of Lemma 5.7. We bound the difference. Similar to Lemma 5.3, we decompose the difference as follows.

$$\begin{aligned}
&\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t) - (\mathbf{J}_{t-1}^{(1)} - \nabla f(\mathbf{x}_{t-1})) \\
&= \nabla f_{\mathcal{S}_t}(\mathbf{x}_t) - \nabla f(\mathbf{x}_t) - \nabla f_{\mathcal{S}_t}(\mathbf{x}_{t-1}) + \nabla f(\mathbf{x}_{t-1}) \\
&\quad + \sum_{s=1}^{p-1}\frac{1}{s!}\bigg\langle \sum_{i=0}^{p-1-s}\frac{1}{i!}\langle \nabla^{s+i+1} f(\widetilde{\mathbf{x}}) - \nabla^{s+i+1} f_{\mathcal{S}_t}(\widetilde{\mathbf{x}}), (\mathbf{x}_{t-1} - \widetilde{\mathbf{x}})^{\otimes i}\rangle, (\mathbf{x}_t - \mathbf{x}_{t-1})^{\otimes s}\bigg\rangle \\
&= \frac{1}{|\mathcal{S}_t|}\sum_{j\in\mathcal{S}_t}[I_t(f_j) - I_t(f)],
\end{aligned} \tag{B.3}$$

where $I_t(g)$ is defined as follows:

$$I_t(g) := \nabla g(\mathbf{x}_t) - \nabla g(\mathbf{x}_{t-1}) - \sum_{s=1}^{p-1}\frac{1}{s!}\bigg\langle \sum_{i=0}^{p-1-s}\frac{1}{i!}\langle \nabla^{s+i+1} g(\widetilde{\mathbf{x}}), (\mathbf{x}_{t-1} - \widetilde{\mathbf{x}})^{\otimes i}\rangle, (\mathbf{x}_t - \mathbf{x}_{t-1})^{\otimes s}\bigg\rangle.$$

We now show an upper bound for $I(g)$ if $\nabla^p g$ is L_p Lipschitz continuous. We have

$$\begin{aligned}
\|I_t(g)\|_2 &\le \underbrace{\bigg\|\sum_{s=1}^{p-1}\frac{1}{s!}\bigg\langle \nabla^{s+1} g(\mathbf{x}_{t-1}) - \sum_{i=0}^{p-1-s}\frac{1}{i!}\langle \nabla^{s+i+1} g(\widetilde{\mathbf{x}}), (\mathbf{x}_{t-1} - \widetilde{\mathbf{x}})^{\otimes i}\rangle, (\mathbf{x}_t - \mathbf{x}_{t-1})^{\otimes s}\bigg\rangle\bigg\|_2}_{J_1} \\
&\quad + \underbrace{\bigg\|\nabla g(\mathbf{x}_t) - \nabla g(\mathbf{x}_{t-1}) - \sum_{s=1}^{p-1}\frac{1}{s!}\langle \nabla^{s+1} g(\mathbf{x}_{t-1}), (\mathbf{x}_t - \mathbf{x}_{t-1})^{\otimes s}\rangle\bigg\|_2}_{J_2},
\end{aligned}$$

For the term J_1, we can bound it as follows:

$$\begin{aligned}
J_1 &\le \sum_{s=1}^{p-1}\frac{1}{s!}\bigg\|\nabla^{s+1} g(\mathbf{x}_{t-1}) - \sum_{i=0}^{p-1-s}\frac{1}{i!}\langle \nabla^{s+i+1} g(\widetilde{\mathbf{x}}), (\mathbf{x}_{t-1} - \widetilde{\mathbf{x}})^{\otimes i}\rangle\bigg\|_{\mathrm{op}}\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2^s \\
&\le \sum_{s=1}^{p-1}\frac{L_p\|\mathbf{x}_{t-1} - \widetilde{\mathbf{x}}\|_2^{p-s}}{s!(p-s)!}\cdot\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2^s
\end{aligned}$$

$$\leq \sum_{s=1}^{p-1} \frac{L_p m^{p-s} r^p}{s!(p-s)!}, \tag{B.4}$$

where the first inequality holds due to the definition of operator norm, the second one holds due to Lemma 3.2, the last one holds due to Lemma 5.1 and the fact that $\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2 = \|\mathbf{h}_{t-1}\|_2 \leq r$. To further bound (B.4), we have

$$\sum_{s=1}^{p-1} \frac{L_p m^{p-s} r^p}{s!(p-s)!} \leq L_p m^{p-1} r^p \sum_{s=1}^{p-1} \frac{1}{s!(p-s)!} \leq L_p m^{p-1} r^p \sum_{s=0}^{p} \frac{1}{s!(p-s)!} = \frac{2^p L_p m^{p-1} r^p}{p!}. \tag{B.5}$$

Substituting (B.5) into (B.4) we obtain the upper bound of J_1. The upper bound of J_2 can be directly obtained by using Lemma 3.2, which is

$$J_2 \leq \frac{L_p}{p!}\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2^p \leq \frac{L_p}{p!} r^p.$$

With the upper bounds for J_1 and J_2, we have

$$\|I_t(g)\|_2 \leq \frac{L_p r^p}{p!}(2(2m)^{p-1} + 1) \leq \frac{3(2m)^{p-1} L_p r^p}{p!}. \tag{B.6}$$

Now we head back to bound $\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t)$. Let q be the index such that $q \leq t < q+m$ and $\widetilde{\mathbf{x}} = \mathbf{x}_q$. Then taking summation of (B.3) from $t' = q+1$ to t, we have

$$\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t) = \sum_{t'=q+1}^{t} [\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t) - (\mathbf{J}_{t-1}^{(1)} - \nabla f(\mathbf{x}_{t-1}))] = \frac{1}{B} \sum_{t'=q+1}^{t} \sum_{j \in \mathcal{S}_{t'}} [I_{t'}(f_j) - I_{t'}(f)].$$

On the one hand, we have $\mathbb{E}_j[I(f_j) - I(f)] = \mathbf{0}$ since $\mathcal{S}_t$ is sampled i.i.d. and $I(g)$ is a linear functional of g. On the other hand, since both $\nabla^p f_j$ and $\nabla^p f$ are L_p Lipschitz continuous, then by (B.6) we have

$$\|I(f_j) - I(f)\|_2 \leq \frac{6(2m)^{p-1} L_p r^p}{p!}.$$

Therefore, by Lemma B.1, for each t, with probability at least $1-\delta$,

$$\|\mathbf{J}_t^{(1)} - \nabla f(\mathbf{x}_t)\|_2 \leq \frac{1}{B} \cdot 2\sqrt{\log(4/\delta)} \cdot \sqrt{(t-q)B} \cdot \frac{6(2m)^{p-1} L_p r^p}{p!} \leq 6\sqrt{\log(4/\delta)} \frac{2^p m^{p-1/2} L_p r^p}{p!\sqrt{B}}.$$

Taking union bound from $t=1$ to T ends our proof. □

Now we begin to prove our main theorems.

B.3. Proof of Theorem 5.4

Proof of Theorem 5.4. We set our parameters as follows.

$$\begin{aligned}
&m = c_1 \bigg(\frac{4^p n}{p^2 \log(4/\delta)}\bigg)^{1/3}, \ B = c_2 \frac{p^{2/3} \log^{2/3}(4/\delta) n^{2/3}}{4^{p/3}}, \\
&M = c_3 \frac{2^{\frac{2p^2+p}{3}} n^{\frac{p-1}{3}} L_p}{p^{\frac{2p-5}{3}} \log^{\frac{p-1}{3}}(4/\delta)}, \ r = c_4 \frac{[(p-1)!]^{1/p} p^{(2p-2)/(3p)} \log^{(p-1)/(3p)}(4/\delta) \epsilon^{1/p}}{L_p^{1/p} 4^{p/3} n^{(p-1)/(3p)}},
\end{aligned} \tag{B.7}$$

where c_i are positive constants. Then it is easy to see there exist c_i to let these parameters satisfy the following conditions:

$$m \cdot B = n, \tag{B.8}$$

$$\frac{M}{8(p+1)!} r^{p+1} = 4\sqrt{\log(4/\delta)} \frac{L_p m^p r^{p+1}}{\sqrt{B} p!}, \ \frac{M}{8(p+1)!} r^{p+1} = \frac{L_p m^{p-1} 2^p r^{p+1}}{(p-1)!}, \tag{B.9}$$

$$4Mr^p/p! = \epsilon. \tag{B.10}$$

First we show that Algorithm 1 indeed ends in T iterations with probability $1 - T\delta$. Let the event defined in Lemma 5.3 holds. Then for any t where Algorithm 1 does not end, by Lemma 4.1, the function value f decreases as follows:

$$\begin{aligned}
f(\mathbf{x}_{t+1}) &\le f(\mathbf{x}_t) - \frac{M}{2(p+1)!}r^{p+1} + \sum_{j=1}^{p}\frac{1}{j!}\|\nabla^j f(\mathbf{x}_t) - \mathbf{J}_t^{(j)}\|_{\mathrm{op}} r^j \\
&\le f(\mathbf{x}_t) - \frac{M}{2(p+1)!}r^{p+1} + 4\sqrt{\log(4/\delta)}\frac{L_p m^p r^{p+1}}{\sqrt{B}p!} + \sum_{j=2}^{p}\frac{r^j}{j!}\cdot\frac{L_p m^{p-j+1} r^{p-j+1}}{(p-j)!} \\
&\le f(\mathbf{x}_t) - \frac{M}{2(p+1)!}r^{p+1} + 4\sqrt{\log(4/\delta)}\frac{L_p m^p r^{p+1}}{\sqrt{B}p!} + \frac{L_p m^{p-1} 2^p r^{p+1}}{p!} \\
&\le f(\mathbf{x}_t) - \frac{M}{4(p+1)!}r^{p+1},
\end{aligned} \tag{B.11}$$

where the second inequality holds due to Lemma 5.3 and 5.2, the last one holds due to (B.9). By (B.11) we know that Algorithm 1 will ends in

$$\Delta_f \Big/ \left(\frac{M}{4(p+1)!}r^{p+1}\right) = T \tag{B.12}$$

number of iterations. Second, let $t < T$ be the iteration where Algorithm 2 ends, and we show that $\mathbf{x}_{t+1}$ is indeed an ϵ-stationary point. By Lemma 4.2 we have

$$\begin{aligned}
\|\nabla f(\mathbf{x}_{t+1})\|_2 &\le \frac{2M}{p!}r^p + \sum_{j=1}^{p}\frac{1}{(j-1)!}\|\nabla^j f(\mathbf{x}_t) - \mathbf{J}_t^{(j)}\|_{\mathrm{op}} r^{j-1} \\
&\le \frac{2M}{p!}r^p + 4\sqrt{\log(4/\delta)}\frac{L_p m^p r^p}{\sqrt{B}p!} + \sum_{j=2}^{p}\frac{r^{j-1}}{(j-1)!}\cdot\frac{L_p m^{p-j+1} r^{p-j+1}}{(p-j)!} \\
&\le \frac{2M}{p!}r^p + 4\sqrt{\log(4/\delta)}\frac{L_p m^p r^p}{\sqrt{B}p!} + \frac{L_p m^{p-1} 2^{p-1} r^p}{(p-1)!} \\
&\le \frac{4M}{p!}r^p \\
&= \epsilon,
\end{aligned}$$

where the second inequality holds due to Lemma 5.3 and 5.2, the last one holds due to (B.10). Finally, we count the total IHO oracle calls. For each iteration t that is not divided by m, Algorithm 2 costs B number of IHO calls. For iteration t divided by m, Algorithm 2 costs n number of IHO calls. Thus the final complexity is

$$T/m\cdot n + TB = 2TB = g(p)\cdot\log^{(p+1)/(3p)}(4/\delta)\cdot\frac{\Delta_f L_p^{1/p} n^{(3p-1)/(3p)}}{\epsilon^{(p+1)/p}},$$

where the first equality holds due to (B.8), the second one holds due to the selection of B and T in (B.12). □

B.4. Proof of Theorem 5.8

Proof of Theorem 5.8. We set our parameters as follows. Let $m \ge p\sqrt{n}$ and

$$B = n/m,\ M = c_1\frac{2^p(p+1)\sqrt{\log(4/\delta)}L_p m^p}{\sqrt{n}},\ r = c_2\frac{(p!)^{1/p} n^{1/(2p)}\epsilon^{1/p}}{(p+1)^{1/p}\log^{1/(2p)}(4/\delta) m L_p^{1/p}}, \tag{B.13}$$

where c_i are positive constants. Then it is easy to see there exist c_i to let these parameters satisfy the following conditions:

$$\frac{M}{8(p+1)!}r^{p+1} = 6\sqrt{\log(4/\delta)}\frac{2^p m^{p-1/2} L_p r^{p+1}}{p!\sqrt{B}},\ 6\sqrt{\log(4/\delta)}\frac{2^p m^{p-1/2} L_p r^{p+1}}{p!\sqrt{B}} \ge \frac{L_p m^{p-1} 2^p r^{p+1}}{(p-1)!}, \tag{B.14}$$

$$\frac{4M}{p!}r^p = \epsilon. \tag{B.15}$$

First we show that Algorithm 1 indeed ends in T iterations with probability $1 - T\delta$. Let the event defined in Lemma 5.7 holds. Then for any t where Algorithm 1 does not end, following the proof of Theorem 5.4, we have

$$\begin{aligned} f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t) - \frac{M}{2(p+1)!}r^{p+1} + 6\sqrt{\log(4/\delta)}\frac{2^p m^{p-1/2} L_p r^{p+1}}{p!\sqrt{B}} + \frac{L_p m^{p-1} 2^p r^{p+1}}{p!} \\ &\leq f(\mathbf{x}_t) - \frac{M}{4(p+1)!}r^{p+1}, \end{aligned} \tag{B.16}$$

where the second inequality holds due to (B.14). By (B.16) we know that Algorithm 1 will ends in

$$\Delta_f \bigg/ \bigg(\frac{M}{4(p+1)!}r^{p+1}\bigg) = T \tag{B.17}$$

number of iterations. Second, let $t < T$ be the iteration where Algorithm 2 ends, then by Lemma 4.2 we have

$$\|\nabla f(\mathbf{x}_{t+1})\|_2 \leq \frac{2M}{p!}r^p + 6\sqrt{\log(4/\delta)}\frac{2^p m^{p-1/2} L_p r^p}{p!\sqrt{B}} + \frac{L_p m^{p-1} 2^{p-1} r^p}{(p-1)!} \leq \frac{4M}{p!}r^p = \epsilon,$$

where the second inequality holds due to Lemma 5.7 and 5.2, the last one holds due to (B.15). Finally, we count the total IHO oracle calls. Similar to the proof of Theorem 5.4, the final complexity is

$$T/m \cdot n + TB = 2TB = g(p) \cdot \log^{1/(2p)}(4/\delta) \cdot \frac{\Delta_f L_p^{1/p} n^{(2p-1)/(2p)}}{\epsilon^{(p+1)/p}},$$

where the first equality holds due to $B = n/m$, the second one holds due to the selection of B and T in (B.17). $\square$