## 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 \(\epsilon\)-stationary points with

\[
O\left(\frac{n^{(2p-1)/(2p)} \epsilon^{(p+1)/p}}{p}\right)
\]

where \(n\) is the number of individual functions and each \(f_i : \mathbb{R}^d \rightarrow \mathbb{R}\) 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 \(\epsilon\)-stationary point \(x\), which is defined by

\[
\|\nabla f(x)\|_2 \leq \epsilon.
\]

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 \(\epsilon\)-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 \(\epsilon\) 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 \(\epsilon^{-2}\) barrier, high-order information including Hessian and higher-order derivatives have to be employed, and better dependence on \(\epsilon^{-1}\) 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. Introduction

We study the following nonconvex finite-sum optimization problem:

\[
\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x),
\]

where \(n\) is the number of individual functions and each \(f_i : \mathbb{R}^d \rightarrow \mathbb{R}\) 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 \(\epsilon\)-stationary point \(x\), which is defined by

\[
\|\nabla f(x)\|_2 \leq \epsilon.
\]

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 \(\epsilon\)-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:
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 \( \epsilon \) at the price of dimension dependence.** This line of works improve the dependence both on \( n \) and \( \epsilon \) 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 \( O(\log d \cdot n^{3/5} / \epsilon^{3/2}) \) 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 \( \epsilon \)?*

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 \( \epsilon \)-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.

### Table 1. Comparisons of different methods to find \( \epsilon \)-stationary points w.r.t. their oracle complexity

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
<th>Lipschitz continuity assumption</th>
<th>Dimension-free?</th>
<th>Small batch size?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR</td>
<td>( O\left(\frac{n}{\epsilon^2}\right) )</td>
<td>Second-order</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>SVRC</td>
<td>( O\left(\log d \cdot \frac{n^{3/5}}{\epsilon^{3/2}}\right) )</td>
<td>Second-order</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>STR2</td>
<td>( O\left(\log d \cdot \frac{n^{3/4}}{\epsilon^{3/2}}\right) )</td>
<td>Second-order</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>HR</td>
<td>( O\left(\frac{n}{\epsilon^{(p+1)/p}}\right) )</td>
<td>( p )-th-order</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td><strong>OP-TE</strong></td>
<td>( O\left(n^{(3p-1)/(3p)} / \epsilon^{(p+1)/p}\right) )</td>
<td>( p )-th-order</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td><strong>TP-TE</strong></td>
<td>( O\left(n^{(2p-1)/(2p)} / \epsilon^{(p+1)/p}\right) )</td>
<td>( p )-th-order</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Lower bound</strong></td>
<td>( \Omega\left(n^{(p-1)/(2p)} / \epsilon^{(p+1)/p}\right) )</td>
<td>( p )-th-order</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

---

\(^1\)The only notable exception is the algorithm proposed by Zhang et al. (2018). However, they made an 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 \( d \).
We call a tensor is symmetric if for any $p$ there is a series of works studying complexity. Zhou et al. (2018a) proposed a stochastic variance reduced cubic regularization method (SVRC) for the finite-sum setting that attains $O(n^{1/5} \epsilon^{-3/2})$ second-order oracle complexity. Zhu 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 $O((p+1)/p)$ 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 $O(\epsilon(p+1)/(p-q+1))$ number of iterations. Cartis et al. (2020b) proposed deterministic but exact trust-region methods that find approximate local minimizer within similar $O(\epsilon^{-q+1})$ function evaluations. Bellavia et al. (2021) proposed stochastic trust-region methods that find approximate local minimizer within similar $O(\epsilon^{-q+1})$ function evaluations. Birgin et al. (2020) proposed a computational feasible fourth-order regularization method that finds stationary points within $O(\epsilon^{-q+1})$ number of iterations. Corresponding to above mentioned upper bound results, there are another 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 $O((p+1)/(p-q+1))$ 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 $O((p+1)/p)$ 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 $O(\epsilon^{-3} + \epsilon^{-5}_h)$ 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 $O(\epsilon^{-2})$ 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 non-convex finite-sum optimization, to find stationary points, Reddi et al. (2016); Allen-Zhu & Hazan (2016) showed an $O(n^{1/2}/\epsilon^2)$ 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 et al., 2018c; Wang et al., 2018a; Nguyen et al., 2019; Cutkosky & Orabona, 2019).
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) \( O(x, i) \) returns the following tuple of tensors:

\[
O(x, i) := (\nabla f_i(x), \nabla^2 f_i(x), \ldots, \nabla^p f_i(x)).
\]

Clearly, IHO is the extension of the existing oracles incremental first-order oracle (Agarwal & Bottou, 2015) and second-order oracle (Zhou et al., 2018). For any tensor function \( g : \mathbb{R}^d \rightarrow (\mathbb{R}^d)^p \), we call it \( L \) Lipschitz continuous if for any \( x_1, x_2 \in \mathcal{X}, \|g(x_1) - g(x_2)\|_{op} \leq L\|x - y\|_2 \).

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

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

\[
\left\| \nabla^s g(y + h) - \nabla^s g(y) - \sum_{j=1}^{p-s} \frac{1}{j!} \nabla^{s+j} g(y, h^{\otimes j}) \right\|_{op} \leq \frac{L_p}{(p - s + 1)!} \|h\|^{p-s+1}.
\]

**Proof.** See Appendix A. \( \square \)

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 \( x_0 \), we have \( f(x_0) - \inf_{x \in \mathbb{R}^d} f(x) \leq \Delta f \).

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 \( x_0 \), at iteration \( t \), HR calculates the tuple of derivatives at iteration \( x_t \), which is \( \langle \nabla f(x_t), \ldots, \nabla^p f(x_t) \rangle \). Then HR updates \( x_{t+1} \leftarrow x_t + h_t \), where \( h_t \) can be set as global minimizer of the following \( p \)-th order regularized Taylor expansion of \( f \) at \( x_t \):

\[
m_t : = \sum_{j=1}^{p} \frac{1}{j!} \nabla^j f(x_t)_j + \frac{M}{(p + 1)!} \|h\|_{2}^{p+1},
\]

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

**Theorem 3.5** (Theorem 2.5, Birgin et al. 2017). By properly setting \( M \), HR outputs \( x_T \), satisfying \( \|\nabla f(x_T)\|_2 \leq \epsilon \), where \( T = O(\Delta f L_p^{1/p} \epsilon^{-(p+1)p}) \). Hence, HR finds \( \epsilon \)-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 \( \hat{J}_t^{(j)}, j = 1, \ldots, p \) to approximate its derivatives \( \nabla^j f(x_t) \) at current iteration \( t \). Then Algorithm 1 computes \( h_t \) as the minimizer of the regularized Taylor expansion \( m_t (4.1) \), within the ball \( \|h\|_2 \leq r \), where \( r \) is the constraint radius. Algorithm 1 updates its iteration \( x_{t+1} \leftarrow x_t + h_t \). Now, Algorithm 1 either proceeds to next iteration or returns \( x_{t+1} \), if the norm of \( h_t \) 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 constraint, and returns iterations based on \( \|h\|_2 \), 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 \( \|h_t\|_2 = r \) and

\[
f(x_{t+1}) \leq f(x_t) - \frac{M}{2(p + 1)!} r^{p+1} + \sum_{j=1}^{p} \frac{1}{j!} \|\nabla^j f(x_t) - \hat{J}_t^{(j)}\|_{op} r^j.
\]

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

\[
f(x_{t+1}) - f(x_t) \leq \sum_{j=1}^{p} \frac{1}{j!} \|\nabla^j f(x_t), h^{\otimes j}\|_2 + \frac{L_p}{(p + 1)!} \|h_t\|_{2}^{p+1},
\]
Then we have

$$\text{where the second inequality holds due to the facts}$$

$$\text{Lemma 4.2. Suppose that Algorithm 1 ends at } t \text{-th iteration. Then we have}$$

$$\|\nabla f(x_t+1)\|_2 \leq \frac{2M}{p!} + \sum_{j=1}^{p} \frac{1}{(j-1)!} \|\nabla f(x_t) - J_t(j)\|_{\text{op}} r^{j-1}.$$  

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

$$\lambda_t(\|h_t\|_2 - r) = 0, \quad \text{(Complementary slackness)}$$

$$0 = \nabla m_t(h_t) + \lambda_t(\|h_t\|_2 - r)|_{h_t = h_t}, \quad \text{(Stationarity).}$$

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

$$\left\| \sum_{j=1}^{p} \frac{1}{(j-1)!}(J_t(j), h_t^{(j-1)}) \right\|_2 = \frac{M}{p!} \|h_t\|_2^2.$$
finding of $\epsilon$-stationary points. Without such a constrain, we can not guarantee that the final output iteration is an $\epsilon$-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 $\|h_t\|_2 \leq r$, 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 $h_t$ satisfies $\|h_t\|_2 \leq r$, (4.2) and (4.3). Therefore, if $\|h_t\|_2 = r$, then by (4.3) we have

$$J^1_t + J^2_t h_t + \frac{M}{2} \| h_t \|^2 \| h_t \|_{h_t} = 0$$

$$\Rightarrow h_t = u_t(\lambda_t) := -(J^2_t + Mr/2I + \lambda_t/rI)^tJ^1_t,$$

where $A^t$ denotes the pseudoinverse of $A$. Since $\|h_t\|_2 = r$, 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 $h_t = u_t(\lambda_t)$. 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 $\|h_t\|_2 < r$, 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 $\bar{x}$, which is repeatedly updated as the current iteration $x_t$ 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.

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

**Proof.** By the definition of $\bar{x}$ we know that there exists an $q$ such that $\bar{x} = x_q$ and $q \leq t \leq q + m$. Using the fact that $\bar{x} - x_t = \sum_{i=q}^{t} (x_i - x_{i+1}) = \sum_{i=q}^{t} h_i$, then we have $\|\bar{x} - x_t\|_2 \leq (t-q)r \leq mr$, where the first inequality holds due to triangle inequality, the second one holds since $\|h_i\|_2 \leq r$ and the last one holds since $0 \leq t-q \leq m$. 

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 $S_t$ with cardinality $|S_t| = B$. Then Algorithm 2 sets the derivative estimator $J^{(j)}_t$ for $j = 1$ and $j \geq 2$ separately. **Case I:** $j \geq 2$. Algorithm 2 constructs $J^{(j)}_t$ as the $(p-j)$-th order Taylor series of derivative function $\nabla^j f$ at point $\bar{x}$. 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 $x_t$, since it only uses the stored derivatives $\nabla^j f(\bar{x})$ at $\bar{x}$. 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 \( 2 \leq j \leq p \) and any \( t \), we have
\[
\|J_t^{(j)}\|_{op} \leq \frac{L_p m^{p-j+1} r_{p-j+1}}{(p-j+1)!}.
\]

**Proof.** By Lemma 3.2, we have
\[
\|J_t^{(j)}\|_{op} = \left\| \nabla f(x_t) + \sum_{s=1}^{p-j} \frac{1}{s!} (\nabla^{j+s} f(x)) (x_t - x) \otimes^s \right\|_{op}
\leq \frac{L_p m^{p-j+1} r_{p-j+1}}{(p-j+1)!}.
\]

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 to concentrate the 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 \( \nabla f(x_t) \) by firstly taking the following difference decomposition:
\[
\nabla f(x_t) = \nabla f_{S_1}(x_t) + \nabla f(\tilde{x}) - \nabla f_{S_1}(\tilde{x}) - (\nabla f_{S_1}(x_t) - \nabla f_{S_1}(\tilde{x})) + \nabla f(x_t) - \nabla f(\tilde{x}),
\]
then Algorithm 2 estimates \( I_1 \) with the Taylor series of \( \nabla f_{S_1}(x) \) expanded at \( \tilde{x} \), \( I_2 \) with the Taylor series of \( \nabla f(x) \) expanded at \( \tilde{x} \), that is,
\[
I_1 \approx \sum_{s=1}^{p-j} \frac{1}{s!} (\nabla^{j+s} f_{S_1}(\tilde{x}), (x_t - \tilde{x}) \otimes^s)
\]
\[
I_2 \approx \sum_{s=1}^{p-j} \frac{1}{s!} (\nabla^{j+s} f(\tilde{x}), (x_t - \tilde{x}) \otimes^s).
\]

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 \( S_t \), the estimator error of the \( J_t^{(1)} \) is smaller by a factor of \( \sqrt{B} \), compared with the previous high-order estimators.

**Lemma 5.3.** Let \( J_t^{(1)} \) be the estimate generated by Algorithm 2. Then with probability at least \( 1 - T \delta \), we have
\[
\|J_t^{(1)} - \nabla f(x_t)\|_{op} \leq 4 \sqrt{\log(4/\delta)} \frac{L_p m^{p-p}}{\sqrt{B p!}}.
\]

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

**Theorem 4.** There exist functions \( g_i, i = 1 \ldots 4 \) that depend only on \( p \) such that, by setting \( m, B, M, r \) as follows:
\[
m = \frac{g_1(p)n^{1/3}}{\log^{1/3}(4/\delta)}, \quad B = g_2(p) \log^{2/3}(4/\delta)n^{2/3},
\]
\[
M = \frac{g_3(p)n^{1/3}}{\log^{1/3}(4/\delta)}, \quad r = \frac{g_4(p) \log^{(p-1)/(3p)}(4/\delta)e^{1/p}}{L_p^{1/p} n^{(p-1)/(3p)}},
\]
and set
\[
T = \Delta_f \left( \frac{M}{4(p+1)!^{p+1}} \right),
\]

### Algorithm 3 TP-TE

**Require:** Reference point update frequency \( m \),
1: if \( t = 0 \mod m \) then
2: \( J_t^{(j)} \leftarrow \nabla_j f(x_t), \forall 1 \leq j \leq p, \tilde{x} \leftarrow x_t \)
3: else
4: Sample \( S_t \in [n], |S_t| = B \)
5: We set \( J_t^{(j)} \) as
6: end if
then with probability at least $1 - T\delta$, Algorithm 1 with Algorithm 2 ends in $T$ steps and outputs an $\epsilon$-stationary point. Meanwhile, there exists a function $g$ that only depends on $p$ such that the number of total IHO calls is bounded as 

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

**Remark 5.5.** Regarding $p$, $\delta$, $\Delta_f$, $L_p$ as constants, Theorem 5.4 suggests an $O(n^{(3p-1)/(3p)} / (p+1)/p)$ dimension-free oracle complexity for Algorithm 2. Compared with HR (Birgin et al., 2017), our result matches its dependence on $\epsilon$, 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 $\bar{x}$ with a $m$-step update frequency. Algorithm 3 only computes the full derivative $\nabla f(x)$ when updating the reference points. At other iterations, it utilizes function information $f_i$, $i \in S_i$ where $|S_i| = B$ 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:

\[
\nabla f(x_t) = \nabla f(x_{t-1}) + \nabla f_{S_t}(x_t) - \nabla f_{S_t}(x_{t-1}) - (\nabla f_{S_t}(x_t) - \nabla f_{S_t}(x_{t-1})) + \nabla f(x_t) - \nabla f(x_{t-1}).
\]

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

\[
J^{(1)}_t \leftarrow J^{(1)}_{t-1} + \nabla f_{S_t}(x_t) - \nabla f_{S_t}(x_{t-1}) - (\nabla f_{S_t}(x_t) - \nabla f_{S_t}(x_{t-1})) + \nabla f(x_t) - \nabla f(x_{t-1}).
\]

To estimate $I_1$ and $I_2$, a natural way is to replace them with the Taylor series of $\nabla f_{S_t}(\bar{x})$ and $\nabla f(\bar{x})$ at point $x_{t-1}$. Since $x_{t-1}$ is 'closer' to the current iteration $x_t$ compared with $\bar{x}$, 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 $\nabla^j f(x_{t-1})$. To work around this issue, instead of using $\nabla^j f(x_{t-1})$, Algorithm 3 computes its $(p - j)$-th order Taylor series expanded at $\bar{x}$ as a replacement, that is, 

\[
\nabla^j f(x_{t-1}) \approx \nabla^j f(\bar{x}) + \sum_{i=1}^{p-j} \frac{1}{i!} (\nabla^{j+i} f(\bar{x}), (x_t - \bar{x})^\otimes i).
\]

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

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

\[
||J^{(1)}_t - \nabla f(x_t)||_2 \leq 6\sqrt{\log(4/\delta) 2^p m^{p-1/2} L_p p^p}.
\]

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

\[
B = n/m, \quad M = g_1(p) \sqrt{\log(4/\delta) L_p m^p / \sqrt{n}},
\]

\[
r = g_2(p) \frac{n^{1/(2p)} \epsilon^{1/p}}{\log^{1/(2p)}(4/\delta) m L_p^{1/p}},
\]

and set

\[
T = \Delta f \left( \frac{M}{4(p+1)!} \right)^{p+1},
\]

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

\[
g(p) \cdot \log^{1/(2p)}(4/\delta) \cdot \Delta_f L_p^{1/p} n^{(2p-1)/(2p)} e^{(p+1)/p}.
\]

**Remark 5.9.** Regarding $p$, $\delta$, $\Delta_f$, $L_p$ as constants, Theorem 5.8 suggests an $O(n^{(2p-1)/(2p)} / (p+1)/p)$ dimension-free oracle complexity for Algorithm 3. Compared with Algorithm 2, the complexity of Algorithm 3 also has an optimal dependence on $\epsilon$. 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 $d$. 

---

**Dimension-free Complexity Bounds for High-order Nonconvex Finite-sum Optimization**
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 \( \epsilon \)-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, \( E||f_i(x) - f_i(y)||_2^2 \leq \ell_2^2\|x - y\|_2^2. \) 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 \( \epsilon \)-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**


A. Proof of Lemma 3.2

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

\[
\nabla^s f(y + h) - \nabla^s f(y) - \sum_{j=1}^{p-s} \frac{1}{j!}(\nabla^{s+j} f(y), h^{\otimes j})
\]

\[
= \int_0^1 \langle \nabla^{s+1} f(y + \lambda h), h \rangle d\lambda - \langle \nabla^{s+1} f(y), h \rangle - \sum_{j=1}^{p-s-1} \frac{1}{(j+1)!}(\nabla^{s+1+j} f(y), h^{\otimes (j+1)})
\]

\[
= \left\langle h, \int_0^1 \nabla^{s+1} f(y + \lambda h) d\lambda - \nabla^{s+1} f(y) - \sum_{j=1}^{p-s-1} \frac{1}{j!}(\nabla^{s+1+j} f(y), h^{\otimes j}) \right\rangle d\lambda,
\]

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

\[
\left\| \nabla^s f(y + h) - \nabla^s f(y) - \sum_{j=1}^{p-s} \frac{1}{j!}(\nabla^{s+j} f(y), h^{\otimes j}) \right\|_{op}
\]

\[
\leq \|h\|_2 \int_0^1 \left\| \nabla^{s+1} f(y + \lambda h) - \nabla^{s+1} f(y) - \sum_{j=1}^{p-s-1} \frac{1}{j!}(\nabla^{s+1+j} f(y), h^{\otimes j}) \right\|_{op} d\lambda
\]

\[
\leq \|h\|_2 \int_0^1 \frac{L_p \lambda^{p-s}}{(p-s)!} \|h\|^{p-s} d\lambda
\]

\[
= \frac{L_p}{(p-s+1)!} \|h\|^{p-s+1}.
\]

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 \( \epsilon_{1:k} \in \mathbb{R}^d \) be a vector-valued martingale difference sequence with respect to \( \mathcal{F}_k \), i.e., for each \( k \in [K] \), \( \mathbb{E}[\epsilon_k | \mathcal{F}_k] = 0 \) and \( \|\epsilon_k\|_2 \leq B_k \), then we have given \( \delta \in (0, 1) \), w.p. \( 1 - \delta \),

\[
\left\| \sum_{i=1}^K \epsilon_k \right\|_2^2 \leq 4 \log(4/\delta) \sum_{i=1}^K B_k^2.
\]

**B.1. Proof of Lemma 5.3**

**Proof of Lemma 5.3.** We bound the difference between \( J_t^{(1)} \) and \( \nabla f(x_t) \). First, we decompose \( J_t^{(1)} - \nabla f(x_t) \) as follows:

\[
J_t^{(1)} - \nabla f(x_t)
\]

\[
= \nabla f_{S_t}(x_t) - \nabla f_{S_t}(\bar{x}) + \nabla f(\bar{x}) - \nabla f(x_t) + \sum_{s=1}^{p-1} \frac{1}{s!}(\nabla^{s+1} f(\bar{x}) - \nabla^{s+1} f_{S_t}(\bar{x}), (x_t - \bar{x})^{\otimes s})
\]

\[
= \frac{1}{|S_t|} \sum_{i \in S_t} \left[ \nabla f_i(x_t) - \nabla f_i(\bar{x}) + \nabla f(\bar{x}) - \nabla f(x_t) + \sum_{s=1}^{p-1} \frac{1}{s!}(\nabla^{s+1} f(\bar{x}) - \nabla^{s+1} f_i(\bar{x}), (x_t - \bar{x})^{\otimes s}) \right] \quad \text{(B.1)}
\]
On the one hand, since $S_t$ is sampled from $[n]$ i.i.d., then we have $E[I_t] = 0$. On the other hand, $I_t$ has the following upper bound:

$$
\|I_t\|_2 \leq \left\| \nabla f_t(x_t) - \nabla f_t(\tilde{x}) - \sum_{s=1}^{p-1} \frac{1}{s!} (\nabla^{s+1} f_t(\tilde{x}), (x_t - \tilde{x})^{\otimes s}) \right\|_2
$$

$$
+ \left\| \nabla f_t(x_t) - \nabla f(\tilde{x}) - \sum_{s=1}^{p-1} \frac{1}{s!} (\nabla^{s+1} f(\tilde{x}), (x_t - \tilde{x})^{\otimes s}) \right\|_2
$$

$$
\leq \frac{2L_p}{p!}\|x_t - \tilde{x}\|_2^p
$$

$$
\leq \frac{2L_p m^{p-1}}{p!}
$$

(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 $1 - \delta$,

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

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

\[\Box\]

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.

$$
J^{(1)}_t - \nabla f(x_t) - (J^{(1)}_{t-1} - \nabla f(x_{t-1}))
$$

$$
= \nabla f_{S_t}(x_t) - \nabla f(x_t) - \nabla f_{S_t}(x_{t-1}) + \nabla f(x_{t-1})
$$

$$
+ \sum_{s=1}^{p-1} \frac{1}{s!} \left\langle \sum_{i=0}^{p-1-s} \frac{1}{i!} (\nabla^{s+i+1} f(\tilde{x}), (x_{t-1} - \tilde{x})^{\otimes i}), (x_t - x_{t-1})^{\otimes s} \right\rangle
$$

$$
= \frac{1}{|S_t|} \sum_{j \in S_t} [I_t(f_j) - I_t(f)],
$$

(B.3)

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

$$
I_t(g) := \nabla g(x_t) - \nabla g(x_{t-1}) - \sum_{s=1}^{p-1} \frac{1}{s!} \left\langle \sum_{i=0}^{p-1-s} \frac{1}{i!} (\nabla^{s+i+1} g(\tilde{x}), (x_{t-1} - \tilde{x})^{\otimes i}), (x_t - x_{t-1})^{\otimes s} \right\rangle.
$$

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

$$
\|I_t(g)\|_2 \leq \sum_{s=1}^{p-1} \frac{1}{s!} \left\| \nabla^{s+1} g(x_{t-1}) - \sum_{i=0}^{p-1-s} \frac{1}{i!} (\nabla^{s+i+1} g(\tilde{x}), (x_{t-1} - \tilde{x})^{\otimes i}) (x_t - x_{t-1})^{\otimes s} \right\|_2
$$

$$
+ \left\| \nabla g(x_t) - \nabla g(x_{t-1}) - \sum_{s=1}^{p-1} \frac{1}{s!} (\nabla^{s+1} g(x_{t-1}), (x_t - x_{t-1})^{\otimes s}) \right\|_2
$$

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

$$
J_1 \leq \sum_{s=1}^{p-1} \frac{1}{s!} \left\| \nabla^{s+1} g(x_{t-1}) - \sum_{i=0}^{p-1-s} \frac{1}{i!} (\nabla^{s+i+1} g(\tilde{x}), (x_{t-1} - \tilde{x})^{\otimes i}) \right\|_{\op} \|x_t - x_{t-1}\|_2
$$

$$
\leq \sum_{s=1}^{p-1} \frac{L_p}{s!} \|x_{t-1} - \tilde{x}\|_2^{p-s} \|x_t - x_{t-1}\|_2^s
$$
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 \(\|x_t - x_{t-1}\|_2 = \|h_{t-1}\|_2 \leq r\). To further bound (B.4), we have

\[
\sum_{s=1}^{p-1} L_p m^{p-s} m^p \leq L_p \sum_{s=1}^{p-1} \frac{1}{s!} \leq \frac{L_p m^{p-1} m^p}{p!} = \frac{2^p L_p m^{p-1} m^p}{p!}.
\]

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

\[
J_2 \leq \frac{L_p}{p!} \|x_t - x_{t-1}\|_2 \leq \frac{L_p m^p}{p!}.
\]

With the upper bounds for \(J_1\) and \(J_2\), we have

\[
\|I_s(g)\|_2 \leq \frac{L_p m^p}{p!} (2(2m)^{p-1} + 1) \leq \frac{3(2m)^{p-1} L_p m^p}{p!}.
\]

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

\[
J_t^{(1)} - \nabla f(x_t) = \sum_{t'=q+1}^{t} [J_t^{(1)} - \nabla f(x_t) - (J_{t'}^{(1)} - \nabla f(x_{t-1}))] = \frac{1}{B} \sum_{t'=q+1}^{t} \sum_{j \in S_{t'}} [I_{t'}(f_j) - I_{t'}(f)].
\]

On the one hand, we have \(\mathbb{E}_{I} [I(f_j) - I(f)] = 0\) since \(S_t\) 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

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

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

\[
\|J_t^{(1)} - \nabla f(x_t)\|_2 \leq \frac{6(2m)^{p-1} L_p m^p}{p!} \leq 6 \sqrt{\log(4/\delta)} \frac{2^p m^{p-1/2} L_p m^p}{p! \sqrt{B}}.
\]

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.

\[
m = c_1 \left( \frac{4^p n}{p^2 \log(4/\delta)} \right)^{1/3}, \quad B = c_2 \left( \frac{2^{2/3} \log^{2/3}(4/\delta) n^{2/3}}{4p^3} \right),
\]

\[
M = c_3 \left( \frac{2^{2/5} n^{1/5}}{p^{2p+5} \log^{3/4} (4/\delta)} \right)^{1/3}, \quad r = c_4 \left( \frac{((p-1)!)^{1/p} p^{(2p-2)/3p} \log^{(p-1)/(3p)} (4/\delta) e^{1/p}}{L_p m^{p(p-1)/3p} n^{1/(3p)}} \right)^{1/3},
\]

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

\[
m \cdot B = n,
\]

\[
\frac{M}{8(p+1)!} p^{p+1} = 4 \sqrt{\log(4/\delta)} \frac{L_p m^{p(p+1)}}{\sqrt{B} p!}, \quad \frac{M}{8(p+1)!} p^{p+1} = \frac{L_p m^{p-1} 2^p p^{p+1}}{(p-1)!}.
\]
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:

$$
4Mr^p/p! = \epsilon. \quad \text{(B.10)}
$$

$$
f(x_{t+1}) \leq f(x_t) - \frac{M}{2(p+1)!} r^{p+1} + \sum_{j=1}^{p} \frac{1}{j!} \|\nabla f(x_t) - J^{(j)}_t\|_{op} r^j
$$

$$
\leq f(x_t) - \frac{M}{2(p+1)!} r^{p+1} + 4\sqrt{\log(4/\delta)} \frac{L_p m^p r^{p+1}}{\sqrt{B} p!} + \sum_{j=2}^{p} \frac{r^j}{j!} \frac{L_p m^{p-j+1} r^{p-j+1}}{(p-j)!}
$$

$$
\leq f(x_t) - \frac{M}{2(p+1)!} r^{p+1} + 4\sqrt{\log(4/\delta)} \frac{L_p m^p r^{p+1}}{\sqrt{B} p!} + \frac{L_p m^{p-1} 2p^{p+1}}{p!}
$$

$$
\leq f(x_t) - \frac{M}{4(p+1)!} r^{p+1}, \quad \text{(B.11)}
$$

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

$$
\Delta f/\left(\frac{M}{4(p+1)!} r^{p+1}\right) = T \quad \text{(B.12)}
$$

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

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

$$
\leq \frac{2M}{p!} r^p + 4\sqrt{\log(4/\delta)} \frac{L_p m^p r^p}{\sqrt{B} p!} + \sum_{j=2}^{p} \frac{r^j}{(j-1)!} \frac{L_p m^{p-j+1} r^{p-j+1}}{(p-j)!}
$$

$$
\leq \frac{2M}{p!} r^p + 4\sqrt{\log(4/\delta)} \frac{L_p m^p r^p}{\sqrt{B} p!} + \frac{L_p m^{p-1} 2p^{p-1}}{(p-1)!}
$$

$$
\leq \frac{4M}{p!} r^p
$$

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). \hfill \square

**B.4. Proof of Theorem 5.8**

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

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

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

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

$$f(x_{t+1}) \leq f(x_t) - \frac{M}{2(p+1)!} r^{p+1} + 6\sqrt{\log(4/\delta)} \frac{2^p n^p L_p r^{p+1}}{p! \sqrt{B}} + \frac{L_p n^p - 2^p r^{p+1}}{p!} \leq f(x_t) - \frac{M}{4(p+1)!} r^{p+1}, \quad (B.16)$$

where the second inequality holds due to (B.14). By (B.16) we know that Algorithm 1 will ends in $\Delta f/\left(\frac{M}{4(p+1)!} r^{p+1}\right) = T$ number of iterations. Second, let $t < T$ be the iteration where Algorithm 2 ends, then by Lemma 4.2 we have

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

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

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

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).