FASTER GRADIENT METHODS FOR HIGHLY-SMOOTH STOCHASTIC BILEVEL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper studies the complexity of finding an ϵ-stationary point for stochastic bilevel optimization when the upper-level problem is nonconvex and the lower-level problem is strongly convex. Recent work proposed the first-order method, $\mathrm{F^2SA}$, achieving the $\tilde{\mathcal{O}}(\epsilon^{-6})$ upper complexity bound for first-order smooth problems. This is slower than the optimal $\Omega(\epsilon^{-4})$ complexity lower bound in its single-level counterpart. In this work, we show that faster rates are achievable for higher-order smooth problems. We first reformulate $\mathrm{F^2SA}$ as approximating the hyper-gradient with a forward difference. Based on this observation, we propose a class of methods $\mathrm{F^2SA}$-p that uses pth-order finite difference for hyper-gradient approximation and improves the upper bound to $\tilde{\mathcal{O}}(p\epsilon^{-4-2/p})$ for pth-order smooth problems. Finally, we demonstrate that the $\Omega(\epsilon^{-4})$ lower bound also holds for stochastic bilevel problems when the high-order smoothness holds for the lower-level variable, indicating that the upper bound of $\mathrm{F^2SA}$-p is nearly optimal in the region $p = \Omega(\log \epsilon^{-1} / \log\log \epsilon^{-1})$.

1 INTRODUCTION

Many machine learning problems, such as meta-learning (Rajeswaran et al., 2019), hyper-parameter tuning (Bao et al., 2021; Franceschi et al., 2018; Mackay et al., 2019), and adversarial training (Goodfellow et al., 2020) can be abstracted as solving the following bilevel optimization problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^{d_x}} \varphi(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x})), \quad \boldsymbol{y}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{y} \in \mathbb{R}^{d_y}} g(\boldsymbol{x}, \boldsymbol{y}), \tag{1}$$

We call f and g the upper-level and lower-level functions, respectively, and call φ the *hyper-objective*. In this paper, we consider the most common *nonconvex-strongly-convex* setting where $f : \mathbb{R}^{d_x} \to \mathbb{R}$ is smooth and possibly nonconvex, and $g : \mathbb{R}^{d_y} \to \mathbb{R}$ is smooth jointly in $(\boldsymbol{x}, \boldsymbol{y})$ and strongly convex in $\boldsymbol{y}$. Under the lower-level strong convexity assumption, the implicit function theorem indicates the following closed form of the hyper-gradient (Ghadimi & Wang, 2018):

$$\nabla \varphi(\boldsymbol{x}) = \nabla_x f(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x})) - \nabla_{xy}^2 g(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x}))[\nabla_{yy}^2 g(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x}))]^{-1} \nabla_y f(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x})). \tag{2}$$

Following the works in nonconvex optimization (Carmon et al., 2020; 2021; Arjevani et al., 2023), we consider the task of finding an ϵ-stationary point of φ, *i.e.*, a point $\boldsymbol{x} \in \mathbb{R}^{d_x}$ such that $\|\nabla \varphi(\boldsymbol{x})\| \le \epsilon$. Motivated by many real machine learning tasks, we study the stochastic setting, where the algorithms only have access to the stochastic derivative estimators of both f and g.

The first efficient algorithm BSA Ghadimi & Wang (2018) for solving the stochastic bilevel problem leverages both stochastic gradient and Hessian-vector-product (HVP) oracles to find an ϵ-stationary point of $\varphi(\boldsymbol{x})$. Subsequently, Ji et al. (2021) proposed stocBiO by incorporating multiple enhanced designs to improve the complexity. Both BSA and stocBiO require the stochastic Hessian assumption (5) on the lower-level function, which is stronger than the standard SGD assumption.

To avoid estimating HVP oracles, Kwon et al. (2023) proposed the first fully first-order method $\mathrm{F^2SA}$ that works under standard SGD assumptions on both f and g (Assumption 2.1). The main idea is to solve the following penalty problem (Liu et al., 2022; 2023; Shen & Chen, 2023; Shen et al., 2025b; Lu & Mei, 2024):

$$\min_{\boldsymbol{x} \in \mathbb{R}^{d_x}, \boldsymbol{y} \in \mathbb{R}^{d_y}} f(\boldsymbol{x}, \boldsymbol{y}) + \lambda \left(g(\boldsymbol{x}, \boldsymbol{y}) - \min_{\boldsymbol{z} \in \mathbb{R}^{d_y}} g(\boldsymbol{x}, \boldsymbol{z}) \right), \tag{3}$$

where λ is taken to be sufficiently large such that $\lambda = \Omega(\epsilon^{-1})$. If we interpret λ as the Lagrangian multiplier, then Problem (3) can be viewed as the Lagrangian function of the constrained optimization $\min_{\boldsymbol{x}\in\mathbb{R}^{d_x},\boldsymbol{y}\in\mathbb{R}^{d_y}} f(\boldsymbol{x},\boldsymbol{y}), \text{s.t. } g(\boldsymbol{x},\boldsymbol{y}) \le g(\boldsymbol{x},\boldsymbol{y}^*(\boldsymbol{x}))$. Thanks to Danskin's theorem, the gradient of the Problem (3) only involves gradient information. Therefore, F^{2}SA does not require the stochastic Hessian assumptions (5). More importantly, by directly leveraging gradient oracles instead of more expensive HVP oracles, the F^{2}SA is more efficient in practice (Shen et al., 2025a; Xiao & Chen, 2025; Jiang et al., 2025) and is also the only method that can be scaled to 32B sized large language model (LLM) training (Pan et al., 2024).

Kwon et al. (2023) proved that the F^{2}SA method finds an ϵ-stationary point of $\varphi(\boldsymbol{x})$ with $\tilde{\mathcal{O}}(\epsilon^{-3})$ first-order oracle calls in the deterministic case and $\tilde{\mathcal{O}}(\epsilon^{-7})$ stochastic first-order oracle (SFO) calls in the stochastic case. Recently, Chen et al. (2025b) showed the two-time-scale stepsize strategy improves the upper complexity bound of F^{2}SA method to $\tilde{\mathcal{O}}(\epsilon^{-2})$ in the deterministic case, which is optimal up to logarithmic factors. However, the direct extension of their method in the stochastic case leads to the $\tilde{\mathcal{O}}(\epsilon^{-6})$ SFO complexity (Chen et al., 2025b; Kwon et al., 2024a) , which still has a significant gap between the $\Omega(\epsilon^{-4})$ lower bound for SGD (Arjevani et al., 2023). It remains open whether optimal rates for stochastic bilevel problems can be achieved for fully first-order methods.

In this work, we revisit F^{2}SA and interpret it as using forward difference to approximate the hypergradient. Our novel interpretation in turn leads to straightforward algorithm extensions for the F^{2}SA method. Observing that the forward difference used by F^{2}SA only has a first-order error guarantee, a natural idea to improve the error guarantee is to use higher-order finite difference methods. For instance, we know that the central difference has an improved second-order error guarantee. Based on this fact, we can derive the F^{2}SA-2 method that solves the following symmetric penalty problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^{d_x},\boldsymbol{y}\in\mathbb{R}^{d_y}} \frac{1}{2}\left(f(\boldsymbol{x},\boldsymbol{y}) + \lambda g(\boldsymbol{x},\boldsymbol{y}) - \min_{\boldsymbol{z}\in\mathbb{R}^{d_y}} \left(-f(\boldsymbol{x},\boldsymbol{z}) + \lambda g(\boldsymbol{x},\boldsymbol{z})\right)\right). \tag{4}$$

Compared with Eq. (3), this new penalty problem perturbs the lower-level variables $\boldsymbol{y}$ and $\boldsymbol{z}$ in the opposite direction to better cancel out the approximation errors to Problem (1). A similar approach has recently been discovered by Chayti & Jaggi (2024) in the context of meta-learning, but they only show its empirical benefit without rigorous theoretical justifications. In this work, we show that F^{2}SA-2 provably improves the SFO complexity of F^{2}SA from $\tilde{\mathcal{O}}(\epsilon^{-6})$ to $\tilde{\mathcal{O}}(\epsilon^{-5})$ for second-order smooth problems. Moreover, our idea is generalizable for any pth-order smooth problems. It is known in numerical analysis there exists the pth-order central difference that uses p points to construct an estimator to the derivative of a unitary function with pth-order error guarantee, as recalled in Lemma 3.1. Motivated by this fact, we propose the F^{2}SA-p algorithm and show that it enjoys the improved $\tilde{\mathcal{O}}(p\epsilon^{-4-2/p})$ SFO complexity, as formally stated in Theorem 3.1.

To examine the tightness of our upper bounds, we further extend the $\Omega(\epsilon^{-4})$ lower bound for SGD (Arjevani et al., 2023) from single-level optimization to bilevel optimization. Note that existing constructions for bilevel lower bound (Dagréou et al., 2024; Kwon et al., 2024a) do not satisfy all our smoothness conditions in Definition 2.2. We demonstrate in Theorem 4.1 that a fully separable construction for upper- and lower-level variables can immediately yield a valid $\Omega(\epsilon^{-4})$ lower bound for the problem class we study, showing that F^{2}SA-p is optimal up to logarithmic factors when $p = \Omega(\log \epsilon^{-1}/\log\log\epsilon^{-1})$ (see Remark 3.4). We summarize our main results, including both the lower and upper bounds, in Table 1 and discuss open problems in the following.

Open problems. Our upper bounds improve known results for high-order smooth problems, but our result still has a gap between the lower bound for $p = \mathcal{O}(\log\epsilon^{-1}/\log\log\epsilon^{-1})$. Recently, Kwon et al. (2024a) obtained some preliminary results towards closing this gap for $p = 1$, where they showed an $\Omega(\epsilon^{-6})$ lower bound holds under a more adversarial oracle. But it is still open whether their lower bounds can be extended to standard stochastic oracles as they conjectured. Another open problem is the tightness of the condition number dependency shown in Table 1.

Notations. We use $\|\cdot\|$ to denote the Euclidean norm for vectors and the spectral norm for matrices and tensors. We use $\tilde{\mathcal{O}}(\,\cdot\,)$ and $\tilde{\Omega}(\,\cdot\,)$ to hide logarithmic factors in $\mathcal{O}(\,\cdot\,)$ and $\Omega(\,\cdot\,)$. We also use $h_1 \lesssim h_2$ to mean $h_1 = \mathcal{O}(h_2)$, $h_1 \gtrsim h_2$ to mean $h_1 = \Omega(h_2)$, and $h_1 \asymp h_2$ to mean that both $h_1 \lesssim h_2$ and $h_1 \gtrsim h_2$ hold. Additional notations for tensors are introduced in Appendix A.

Method	Smoothness	Reference	Complexity
F^{2}SA	1st-order	(Kwon et al., 2023)	$\tilde{\mathcal{O}}(\mathrm{poly}(\kappa)\epsilon^{-7})$
F^{2}SA	1st-order	(Kwon et al., 2024a)	$\tilde{\mathcal{O}}(\mathrm{poly}(\kappa)\epsilon^{-6})$
F^{2}SA	1st-order	(Chen et al., 2025b)	$\tilde{\mathcal{O}}(\kappa^{12}\epsilon^{-6})$
F^{2}SA-p	1st-order + pth-order in $\boldsymbol{y}$	Theorem 3.1	$\tilde{\mathcal{O}}(p\kappa^{9+2/p}\epsilon^{-4-2/p})$
Lower Bound		Theorem 4.1	$\Omega(\epsilon^{-4})$

Table 1: The SFO complexity of different methods to find an ϵ-stationary point for pth-order smooth first-order bilevel problems with condition number κ under standard SGD assumptions.

2 Preliminaries

The goal of bilevel optimization is to minimize the hyper-objective $\varphi(\boldsymbol{x})$, which is in general nonconvex. Since finding a global minimizer of a general nonconvex function requires exponential complexity in the worst case (Nemirovskij & Yudin, 1983, § 1.6), we follow the literature (Carmon et al., 2020; 2021) to consider the task of finding an approximate stationary point.

Definition 2.1. *Let $\varphi : \mathbb{R}^{d_x} \to \mathbb{R}$ be the hyper-objective defined in Eq. (1). We say a random variable $\hat{\boldsymbol{x}} \in \mathbb{R}^{d_x}$ is an ϵ-hyper-stationary point if $\mathbb{E}\|\nabla\varphi(\hat{\boldsymbol{x}})\| \le \epsilon$.*

Next, we introduce the assumptions used in this paper, which ensure the tractability of the above hyper-stationarity. Compared to (Kwon et al., 2023; Chen et al., 2025b), we additionally assume the high-order smoothness in lower-level variable $\boldsymbol{y}$ to achieve further acceleration.

2.1 Problem Setup

First of all, we follow the standard assumptions on SGD (Arjevani et al., 2023) to assume that the stochastic gradient estimators satisfy the following assumption.

Assumption 2.1. *There exists stochastic gradient estimators $F(\boldsymbol{x},\boldsymbol{y})$ and $G(\boldsymbol{x},\boldsymbol{y})$ such that*

$$\mathbb{E}F(\boldsymbol{x},\boldsymbol{y};\xi) = \nabla f(\boldsymbol{x},\boldsymbol{y}), \quad \mathbb{E}\|F(\boldsymbol{x},\boldsymbol{y}) - \nabla f(\boldsymbol{x},\boldsymbol{y})\|^2 \le \sigma^2;$$
$$\mathbb{E}G(\boldsymbol{x},\boldsymbol{y};\zeta) = \nabla g(\boldsymbol{x},\boldsymbol{y}), \quad \mathbb{E}\|G(\boldsymbol{x},\boldsymbol{y}) - \nabla g(\boldsymbol{x},\boldsymbol{y})\|^2 \le \sigma^2,$$

where $\sigma > 0$ is the variance of the stochastic gradient estimators. We also partition $F = (F_x, F_y)$ and $G = (G_x, G_y)$ such that F_x, F_y, G_x, G_y are estimators to $\nabla_x f, \nabla_y f, \nabla_x g, \nabla_y g$, respectively.

Second, we assume that the hyper-objective $\varphi(\boldsymbol{x}) = f(\boldsymbol{x},\boldsymbol{y}^*(\boldsymbol{x}))$ is lower bounded. Otherwise, any algorithm requires infinite time to find a stationary point. Note that the implicit condition $\inf_{\boldsymbol{x}\in\mathbb{R}^{d_x}} \varphi(\boldsymbol{x}) > -\infty$ below can also be easily implied by a more explicit condition on the lower boundedness of upper-level function $\inf_{\boldsymbol{x}\in\mathbb{R}^{d_x},\boldsymbol{y}\in\mathbb{R}^{d_y}} f(\boldsymbol{x},\boldsymbol{y}) > -\infty$.

Assumption 2.2. *The hyper-objective defined in Eq. (1) is lower bounded, and we have*

$$\varphi(\boldsymbol{x}_0) - \inf_{\boldsymbol{x}\in\mathbb{R}^{d_x}} \varphi(\boldsymbol{x}) \le \Delta,$$

where $\Delta > 0$ is the initial suboptimality gap and we assume $\boldsymbol{x}_0 = \mathbf{0}$ without loss of generality.

Third, we assume the lower-level function $g(\boldsymbol{x},\boldsymbol{y})$ is strongly convex in $\boldsymbol{y}$. It guarantees the uniqueness of $\boldsymbol{y}^*(\boldsymbol{x})$ and the tractability of the bilevel problem. Although not all the problems in applications satisfy the lower-level strong convexity assumption, it is impossible to derive dimension-free upper bounds when the lower-level problem is only convex (Chen et al., 2024, Theorem 3.2). Hence, we follow most existing works to consider strongly convex lower-level problems.

Assumption 2.3. *$g(\boldsymbol{x},\boldsymbol{y})$ is μ-strongly convex in $\boldsymbol{y}$, i.e., for any $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{R}^{d_y}$, we have*

$$g(\boldsymbol{x},\boldsymbol{y}_2) \ge g(\boldsymbol{x},\boldsymbol{y}_1) + \langle \nabla_y g(\boldsymbol{x},\boldsymbol{y}_1), \boldsymbol{y}_2 - \boldsymbol{y}_1\rangle + \frac{\mu}{2}\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|^2,$$

where $\mu > 0$ is the strongly convex parameter.

Fourth, we require the following smoothness assumptions following (Ghadimi & Wang, 2018). According to Eq. (2), these conditions are necessary and sufficient to guarantee the Lipschitz continuity of $\nabla\varphi(\boldsymbol{x})$, which further ensure the tractability of an approximate stationary point of the nonconvex hyper-objective $\varphi(\boldsymbol{x})$ (Zhang et al., 2020; Kornowski & Shamir, 2022).

Assumption 2.4. *For the upper-lower function f and lower-level function g, we assume that*

1. *$f(\boldsymbol{x},\boldsymbol{y})$ is L_0-Lipschitz in $\boldsymbol{y}$.*
2. *$\nabla f(\boldsymbol{x},\boldsymbol{y})$ and $\nabla g(\boldsymbol{x},\boldsymbol{y})$ are L_1-Lipschitz jointly in $(\boldsymbol{x},\boldsymbol{y})$.*
3. *$\nabla^2_{xy} g(\boldsymbol{x},\boldsymbol{y})$ and $\nabla^2_{yy} g(\boldsymbol{x},\boldsymbol{y})$ are L_2-Lipschitz jointly in $(\boldsymbol{x},\boldsymbol{y})$.*

We refer to the problem class that jointly satisfies all the above Assumption 2.1, 2.2, 2.3 and 2.4 as first-order smooth bilevel problems, for which (Kwon et al., 2024a; Chen et al., 2025b) showed the F^{2}SA method achieves the $\tilde{\mathcal{O}}(\epsilon^{-6})$ upper complexity bound. In this work, we show an improved bound under the following additional higher-order smoothness assumption on lower-level variable $\boldsymbol{y}$.

Assumption 2.5 (High order smoothness in $\boldsymbol{y}$)**.** *Given $p \in \mathbb{N}_+$, we assume that*

1. *$\frac{\partial^q}{\partial \boldsymbol{y}^q}\nabla f(\boldsymbol{x},\boldsymbol{y})$ is L_{q+1}-Lipschitz for all $q = 1,\cdots,p-1$.*
2. *$\frac{\partial^{q+1}}{\partial \boldsymbol{y}^{q+1}}\nabla g(\boldsymbol{x},\boldsymbol{y})$ is L_{q+2}-Lipschitz in $\boldsymbol{y}$ for all $q = 1,\cdots,p-1$.*

We refer to problems jointly satisfying all the above assumptions as pth-order smooth bilevel problems, and also formally define their condition numbers as follows.

Definition 2.2 (pth-order smooth bilevel problems)**.** *Given $p \in \mathbb{N}_+$, $\Delta > 0$, $L_0, L_1, \cdots, L_{p+1} > 0$, and $\mu \le L_1$, we use $\mathcal{F}^{\text{nc-sc}}(L_0,\cdots,L_{p+1},\mu,\Delta)$ to denote the set of all bilevel instances satisfying Assumption 2.2, 2.3, 2.4 and 2.5. For this problem class, we define the largest smoothness constant $\bar{L} = \max_{0\le j\le p} L_j$ and condition number $\kappa = \bar{L}/\mu$.*

All our above assumptions align with (Chen et al., 2025b) except for the additional Assumption 2.5. A classical example of a highly smooth function is the softmax function (Garg et al., 2021, Lemma 2(3)). Therefore, many hyper-parameter tuning problems for logistic regression are provably highly smooth, such that our theory can be applied. We give two examples from (Pedregosa, 2016): the first one aims to learn the optimal weights of each sample in a corrupted training set, and the second one aims to learn the optimal regularizer of each parameter.

Example 2.1 (Data hyper-cleaning)**.** *Let $\boldsymbol{x} \in \mathbb{R}^n$ parameterize the per-sample weight of a training set with n samples via $\sigma(x_i) = \exp(x_i)/\sum_{i=1}^n \exp(x_i)$ and $\boldsymbol{y} \in \mathbb{R}^d$ be the parameters of a linear model. Let ℓ_{val} be the logistic loss of the linear model on the validation set and ℓ^i_{tr} be the logistic loss on the training sample i. The problem aims to solve*

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \ell_{\text{val}}(\boldsymbol{y}), \quad \text{s.t.} \quad \boldsymbol{y} \in \arg\min_{\boldsymbol{y}\in\mathbb{R}^d} \sum_{i=1}^n \sigma(x_i)\ell^i_{\text{tr}}(\boldsymbol{y}).$$

Example 2.2 (Learn-to-regularize)**.** *Let $\boldsymbol{x} \in \mathbb{R}^d$ parameterize the regularization matrix via $\boldsymbol{W}_{\boldsymbol{x}} = \operatorname{diag}(\exp(\boldsymbol{x}))$, and $\boldsymbol{y} \in \mathbb{R}^d$ be the parameters of a linear model. Let ℓ_{val} and ℓ_{tr} be the logistic loss of the linear model on the validation set and training set, respectively. The problem aims to solve*

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \ell_{\text{val}}(\boldsymbol{y}), \quad \text{s.t.} \quad \boldsymbol{y} \in \arg\min_{\boldsymbol{y}\in\mathbb{R}^d} \ell_{\text{tr}}(\boldsymbol{y}) + \boldsymbol{y}^\top \boldsymbol{W}_{\boldsymbol{x}}\boldsymbol{y}.$$

2.2 Comparison to Previous Works

Before we show our improved upper bound, we first give a detailed discussion on other assumptions made in related works.

Stochastic Hessian assumption. Ghadimi & Wang (2018); Ji et al. (2021) assumes the access to a stochastic Hessian estimator $\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y})$ such that

$$\mathbb{E}\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y}) = \nabla^2 g(\boldsymbol{x},\boldsymbol{y}), \quad \mathbb{E}\|\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y}) - \nabla^2 g(\boldsymbol{x},\boldsymbol{y})\| \le \sigma^2. \tag{5}$$

According to (Arjevani et al., 2020, Observation 1 and 2), such an assumption is stronger than standard SGD assumptions and equivalent to the mean-squared-smoothness assumption (6) on the lower-level gradient estimator G under the mild condition of $\nabla G(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{H}(\boldsymbol{x},\boldsymbol{y})$. Under this assumption, in conjunction with Assumption 2.2, 2.3, and 2.4, Ghadimi & Wang (2018) proposed the BSA method that can find an ϵ stationary point of $\varphi(\boldsymbol{x})$ with $\tilde{\mathcal{O}}(\epsilon^{-6})$ SFO complexity and $\tilde{\mathcal{O}}(\epsilon^{-4})$ stochastic HVP complexity. Later, Ji et al. (2021) further improved the SFO complexity term to $\tilde{\mathcal{O}}(\epsilon^{-4})$. Compared to them, we consider the setting where the algorithms only have access to stochastic gradient estimators, and make no assumptions on the stochastic Hessians.

Mean-squared smoothness assumption. Besides Assumption 2.1, 2.2, 2.3, 2.4 and the stochastic Hessian assumption (5), Khanduri et al. (2021); Yang et al. (2021; 2023b) further assumes that the stochastic estimators to gradients and Hessians are mean-squared smooth:

$$\begin{aligned} \mathbb{E}\|F(\boldsymbol{x},\boldsymbol{y}) - F(\boldsymbol{x}',\boldsymbol{y}')\|^2 &\le \bar{L}_1^2\|(\boldsymbol{x},\boldsymbol{y}) - (\boldsymbol{x}',\boldsymbol{y}')\|^2, \\ \mathbb{E}\|G(\boldsymbol{x},\boldsymbol{y}) - G(\boldsymbol{x}',\boldsymbol{y}')\|^2 &\le \bar{L}_1^2\|(\boldsymbol{x},\boldsymbol{y}) - (\boldsymbol{x}',\boldsymbol{y}')\|^2, \\ \mathbb{E}\|\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y}) - \boldsymbol{H}(\boldsymbol{x}',\boldsymbol{y}')\|^2 &\le \bar{L}_2^2\|(\boldsymbol{x},\boldsymbol{y}) - (\boldsymbol{x}',\boldsymbol{y}')\|^2. \end{aligned} \tag{6}$$

Under this additional assumption, they proposed faster stochastic methods with upper complexity bound of $\tilde{\mathcal{O}}(\epsilon^{-3})$ via variance reduction (Fang et al., 2018; Cutkosky & Orabona, 2019). In this paper, we only consider the setting without mean-squared smoothness assumptions and study a different acceleration mechanism from variance reduction.

Jointly high-order smoothness assumption. Huang et al. (2025) introduced a second-order smoothness assumption similar to but stronger than Assumption 2.5 when $p = 2$. Specifically, they assumed the second-order smoothness jointly in $(\boldsymbol{x},\boldsymbol{y})$ instead of $\boldsymbol{y}$ only:

$$\begin{aligned} &\nabla^2 f(\boldsymbol{x},\boldsymbol{y}) \text{ is } L_2\text{-Lipschitz jointly in } (\boldsymbol{x},\boldsymbol{y}); \\ &\nabla^3 g(\boldsymbol{x},\boldsymbol{y}) \text{ is } L_3\text{-Lipschitz jointly in } (\boldsymbol{x},\boldsymbol{y}). \end{aligned} \tag{7}$$

The jointly second-order smoothness (7) ensures that the hyper-objective $\varphi(\boldsymbol{x})$ has Lipschitz continuous Hessians, which further allows the application of known techniques in minimizing second-order smooth objectives. Huang et al. (2025) applied the technique from (Jin et al., 2017; 2021; Xu et al., 2018; Allen-Zhu & Li, 2018) to show that an HVP-based method can find a second-order stationary point in $\tilde{\mathcal{O}}(\epsilon^{-2})$ complexity under the deterministic setting, and in $\tilde{\mathcal{O}}(\epsilon^{-4})$ under the stochastic Hessian assumption (5). Yang et al. (2023a) applied the technique from (Li & Lin, 2023) to accelerate the complexity HVP-based method to $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ in the deterministic setting. Chen et al. (2025b) also proposed a fully first-order method to achieve the same $\tilde{\mathcal{O}}(\epsilon^{-1.75})$ complexity. Compared to these works, our work demonstrates a unique acceleration mechanism in stochastic bilevel optimization that only comes from the high-order smoothness in $\boldsymbol{y}$.

3 THE F^{2}SA-p METHOD

To introduce our method, we first recall the prior F^{2}SA method (Kwon et al., 2023) and establish their relationship between finite difference schemes, which further motivates us to design better algorithms by using better finite difference formulas.

3.1 HYPER-GRADIENT APPROXIMATION VIA FINITE DIFFERENCE

The core idea of F^{2}SA (Kwon et al., 2023) is to solve the reformulated penalty problem (3) and use the gradient of the penalty function to approximate the true hyper-gradient. To make connections of F^{2}SA to the finite difference method, let us introduce the extra notation g_ν as the perturbed lower-lever problem with $\boldsymbol{y}_\nu^*(\boldsymbol{x})$ and $\ell_\nu(\boldsymbol{x})$ being its optimal solution and optimal value, respectively:

$$\begin{aligned} g_\nu(\boldsymbol{x},\boldsymbol{y}) &:= \nu f(\boldsymbol{x},\boldsymbol{y}) + g(\boldsymbol{x},\boldsymbol{y}), \\ \boldsymbol{y}_\nu^*(\boldsymbol{x}) &:= \arg\min_{\boldsymbol{y}\in\mathbb{R}^{d_y}} g_\nu(\boldsymbol{x},\boldsymbol{y}), \\ \ell_\nu(\boldsymbol{x}) &:= \min_{\boldsymbol{y}\in\mathbb{R}^{d_y}} g_\nu(\boldsymbol{x},\boldsymbol{y}), \end{aligned}$$

Then we have $\frac{\partial}{\partial \nu}\ell_\nu(\boldsymbol{x})|_{\nu=0} = \lim_{\nu\to 0}\frac{\ell_\nu(\boldsymbol{x})-\ell_0(\boldsymbol{x})}{\nu} = \lim_{\nu\to 0} f(\boldsymbol{x},\boldsymbol{y}_\nu^*(\boldsymbol{x})) + \frac{g(\boldsymbol{x},\boldsymbol{y}_\nu^*(\boldsymbol{x}))-g(\boldsymbol{x},\boldsymbol{y}^*(\boldsymbol{x}))}{\nu}$. In our notation, we can rewrite (Chen et al., 2025b, Lemma B.3) as $\frac{\partial}{\partial \nu}\ell_\nu(\boldsymbol{x})|_{\nu=0} = \varphi(\boldsymbol{x})$. Similarly, we can also rewrite (Kwon et al., 2023, Lemma 3.1) as

$$\frac{\partial^2}{\partial \nu \partial \boldsymbol{x}}\ell_\nu(\boldsymbol{x})|_{\nu=0} = \frac{\partial^2}{\partial \boldsymbol{x} \partial \nu}\ell_\nu(\boldsymbol{x})|_{\nu=0} = \nabla\varphi(\boldsymbol{x}). \tag{8}$$

Let $\nu = 1/\lambda$ in Eq. (3). Then the fully first-order hyper-gradient estimator (Kwon et al., 2023; Chen et al., 2025b) is exactly using forward difference to approximate $\nabla\varphi(\boldsymbol{x})$, that is,

$$\frac{\frac{\partial}{\partial \boldsymbol{x}}\ell_\nu(\boldsymbol{x}) - \frac{\partial}{\partial \boldsymbol{x}}\ell_0(\boldsymbol{x})}{\nu} \approx \frac{\partial^2}{\partial \nu \partial \boldsymbol{x}}\ell_\nu(\boldsymbol{x})|_{\nu=0} = \nabla\varphi(\boldsymbol{x}). \tag{9}$$

However, the forward difference is not the only way to approximate a derivative. Essentially, it falls into a general class of pth-order finite difference (Atkinson & Han, 2005) that can guarantee an $\mathcal{O}(\nu^p)$ approximation error. We restate this known result (with generalization to vector-valued functions) in the following lemma and provide a self-contained proof in Appendix B for completeness.

Lemma 3.1. *Assume the function $\psi : \mathbb{R} \to \mathbb{R}^d$ has C-Lipschitz continuous pth-order derivative. There exist coefficients $\{\alpha_j\}$ such that*

$$\left\| \frac{1}{\nu}\sum_j \alpha_j \psi(j\nu) - \psi'(0) \right\| = \mathcal{O}(C\nu^p).$$

If p is even, the indices run $j = -p/2, \cdots, p/2$. If p is odd, they run $j = -(p-1)/2, \cdots, (p+1)/2$. Furthermore, all the coefficients satisfy $|j\alpha_j| \le 1$ for all $j \neq 0$ and $|\alpha_0| \le \mathbb{I}[p \text{ is odd}]$.

The explicit formulas for α_j can be found in Appendix B. When $p = 1$, we have $\alpha_0 = -1, \alpha_1 = 1$, and we obtain the forward difference estimator $\psi(\nu) - \psi(0)/\nu$; When $p = 2$ we have $\alpha_{-1} = -1/2, \alpha_1 = 1/2$ and we obtain the central difference estimator $(\psi(\nu) - \psi(-\nu))/(2\nu)$. Lemma 3.1 tells us that in general we can always construct a finite difference estimator $\mathcal{O}(\nu^p)$ error with p points for even p or $p+1$ points for odd p under the given smoothness conditions. Inspired by Lemma 3.1 and Eq. (8) that $\frac{\partial^2}{\partial \nu \partial \boldsymbol{x}}\ell_\nu(\boldsymbol{x})|_{\nu=0} = \nabla\varphi(\boldsymbol{x})$, we propose a fully first-order estimator via a linear combination of $\frac{\partial}{\partial \boldsymbol{x}}\ell_{j\nu}(\boldsymbol{x})$ to achieve $\mathcal{O}(\nu^p)$ approximation error to $\nabla\varphi(\boldsymbol{x})$ given that $\frac{\partial^{p+1}}{\partial \nu^p \partial \boldsymbol{x}}\ell_\nu(\boldsymbol{x})$ is Lipschitz continuous in ν. It further leads to Algorithm 1 that will be formally introduced in the next subsection.

3.2 The Proposed Algorithm

Due to space limitations, we only present Algorithm 1 designed for even p in the main text. The algorithm for odd p can be designed similarly, and we defer the concrete algorithm to Appendix D.

Algorithm 1 follows the double-loop structure of F^{2}SA (Chen et al., 2025b; Kwon et al., 2024a) and changes the hyper-gradient estimator to the one introduced in the previous section. Now, we give a more detailed introduction to the procedures of the two loops of F^{2}SA-p.

1. In the outer loop, the algorithm first samples a mini-batch with size S and uses Lemma 3.1 to construct Φ_t via the linear combination of $\frac{\partial}{\partial \boldsymbol{x}}\ell_{j\nu}(\boldsymbol{x}_t)$ for $j = -p/2, \cdots, p/2$ every iteration. After obtaining Φ_t as an approximation to $\nabla\varphi(\boldsymbol{x}_t)$, the algorithm then performs a normalized gradient descent step $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_x \Phi_t/\|\Phi_t\|$ with total T iterations.
2. In the inner loop, the algorithm returns an approximation to $\frac{\partial}{\partial \boldsymbol{x}}\ell_{j\nu}(\boldsymbol{x}_t)$ for all $j = -p/2, \cdots, p/2$. Note that Danskin's theorem indicates $\frac{\partial}{\partial \boldsymbol{x}}\ell_{j\nu}(\boldsymbol{x}_t) = \frac{\partial}{\partial \boldsymbol{x}}g_{j\nu}(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t))$. It suffices to approximate $\boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)$ to sufficient accuracy, which is achieved by taking a K-step single-batch SGD subroutine with stepsize η_y on each function $g_{j\nu}(\boldsymbol{x}, \cdot)$.

Remark 3.1 (Effect of normalized gradient step)**.** *Compared to (Chen et al., 2025b; Kwon et al., 2023), the only modification we make to the outer loop is to change the gradient step to a normalized gradient step. The normalization can control the change of $\boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)$ and make the analysis of inner loops easier. We believe that all our theoretical guarantees also hold for the standard gradient step via a more involved analysis.*

Algorithm 1 $\mathrm{F}^2\mathrm{SA}\text{-}p\ (\boldsymbol{x}_0, \boldsymbol{y}_0)$, even p

$\boldsymbol{y}_0^j = \boldsymbol{y}_0,\ \forall j \in \mathbb{N}$
for $t = 0, 1, \cdots, T-1$
  parallel for $j = -p/2, -p/2+1, \cdots, p/2$
    $\boldsymbol{y}_t^{j,0} = \boldsymbol{y}_t^j$
    for $k = 0, 1, \cdots, K-1$
      Sample random i.i.d indexes $\{(\xi_j^y, \zeta_j^y)\}$.
      $\boldsymbol{y}_t^{j,k+1} = \boldsymbol{y}_t^{j,k} - \eta_y \left(j\nu F_y(\boldsymbol{x}_t, \boldsymbol{y}_t^{j,k}; \xi_j^y) + G_y(\boldsymbol{x}_t, \boldsymbol{y}_t^{j,k}; \zeta_j^y) \right)$
    end for
    $\boldsymbol{y}_{t+1}^j = \boldsymbol{y}_t^{j,K}$
  end parallel for
  Sample random i.i.d indexes $\{(\xi_i^x, \zeta_i^x)\}_{i=1}^S$.
  Let $\{\alpha_j\}_{j=-p/2}^{p/2}$ be the pth-order finite difference coefficients defined in Lemma 3.1.
  $\Phi_t = \frac{1}{S} \sum_{i=1}^S \sum_{j=-p/2}^{p/2} \alpha_j \left(jF_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j; \xi_i^x) + \dfrac{G_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j; \zeta_i^x)}{\nu} \right)$
  $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_x \Phi_t / \|\Phi_t\|$
end for

3.3 Complexity Analysis

This section contains the complexity analysis of Algorithm 1. We first derive the following lemma from the high-dimensional Faà di Bruno formula (Licht, 2024).

Lemma 3.2. *Let $\nu \in (0, 1/(2\kappa)]$. For any instance in the pth-order smooth bilevel problem class $\mathcal{F}^{\text{nc-sc}}(L_0, \cdots, L_{p+1}, \mu, \Delta)$ as Definition 2.2, $\frac{\partial^{p+1}}{\partial \nu^p \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2p+1}\bar{L})$-Lipschitz continuous in ν.*

Our result generalizes the prior result for $p = 1$ (Kwon et al., 2023) to any $p \in \mathbb{N}_+$ and also tightens the prior bounds for $p = 2$ (Chen et al., 2025b) as we remark in the following.

Remark 3.2 (Tighter bounds for $p = 2$)**.** *Note that the variables $\boldsymbol{x}$ and ν play equal roles in our analysis. Therefore, our result in $p = 2$ essentially implies that $\frac{\partial^3}{\partial \nu \partial \boldsymbol{x}^2} \ell_\nu(\boldsymbol{x})$ is $\mathcal{O}(\kappa^5 \bar{L})$-Lipschitz continuous in ν around zero, which tightens the $\mathcal{O}(\kappa^6 \bar{L})$ bound of Hessian convergence in (Chen et al., 2025b, Lemma 5.1a) and is of independent interest. The main insight is to avoid the direct calculation of $\nabla^2 \varphi(\boldsymbol{x}) = \frac{\partial^3}{\partial \nu \partial \boldsymbol{x}^2} \ell_\nu(\boldsymbol{x})|_{\nu=0}$ which involves third-order derivatives and makes the analysis more complex, but instead always to analyze it through the limiting point $\lim_{\nu \to 0} \frac{\partial^3}{\partial \nu \partial \boldsymbol{x}^2} \ell_\nu(\boldsymbol{x})$.*

Recall Eq. (8) that $\frac{\partial^2}{\partial \nu \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})|_{\nu=0} = \nabla \varphi(\boldsymbol{x})$. Then Lemma 3.2, in conjunction with Lemma 3.1, indicates that the pth-order finite difference used in $\mathrm{F}^2\mathrm{SA}\text{-}p$ guarantees an $\mathcal{O}(\nu^p)$-approximation error to $\nabla \varphi(\boldsymbol{x})$, which always improves the $\mathcal{O}(\nu)$-error guarantee of $\mathrm{F}^2\mathrm{SA}$ (Kwon et al., 2023; Chen et al., 2025b) for any $p \geq 2$. This improved error guarantee means that we can set $\nu = \mathcal{O}(\epsilon^{1/p})$ to obtain an $\mathcal{O}(\epsilon)$-accurate hyper-gradient estimator to $\nabla \varphi(\boldsymbol{x})$, which further leads to the following improved complexity of our algorithm.

Theorem 3.1 (Main theorem)**.** *For any instance in the pth-order smooth bilevel problem class $\mathcal{F}^{\text{nc-sc}}(L_0, \cdots, L_{p+1}, \mu, \Delta)$ as per Definition 2.2, set the hyper-parameters as*

$$\begin{aligned} \nu &\asymp \min \left\{ \frac{R}{\kappa}, \left(\frac{\epsilon}{\bar{L}\kappa^{2p+1}} \right)^{1/p} \right\}, \quad \eta_x \asymp \frac{\epsilon}{L_1 \kappa^3}, \quad \eta_y \asymp \frac{\nu^2 \epsilon^2}{L_1 \kappa \sigma^2}, \\ S &\asymp \frac{\sigma^2}{\nu^2 \epsilon^2}, \quad K \asymp \frac{\kappa^2 \sigma^2}{\nu^2 \epsilon^2} \log \left(\frac{R L_1 \kappa}{\nu \epsilon} \right), \quad T \asymp \frac{\Delta}{\eta_x \epsilon}, \end{aligned} \tag{10}$$

where $R = \|\boldsymbol{y}_0 - \boldsymbol{y}^*(\boldsymbol{x}_0)\|$. Run Algorithm 1 if p is even or Algorithm 2 (in Appendix D) if p is odd. Then we can provably find an ϵ-stationary point of $\varphi(\boldsymbol{x})$ with the total SFO calls upper bounded by

$$pT(S+K) = \mathcal{O}\left(\frac{p \Delta L_1 \bar{L}^{2/p} \sigma^2 \kappa^{9+2/p}}{\epsilon^{4+2/p}} \log \left(\frac{R L_1 \bar{L} \kappa}{\epsilon} \right) \right).$$

The above theorem shows that the F^{2}SA-p method can achieve the $\tilde{\mathcal{O}}(p\kappa^{9+2/p}\epsilon^{-4-2/p}\log(\kappa/\epsilon))$ SFO complexity for pth-order smooth bilevel problems. In the following, we give several remarks on the complexity in different regions of p.

Remark 3.3 (First-order smooth region)**.** *For $p=1$, our upper bound becomes $\tilde{\mathcal{O}}(\kappa^{11}\epsilon^{-6})$, which improves the $\tilde{\mathcal{O}}(\kappa^{12}\epsilon^{-6})$ bound in (Chen et al., 2025b) by a factor of κ. The improvement comes from a tighter analysis in the lower-level SGD update and a careful parameter setting.*

Remark 3.4 (Highly-smooth region)**.** *For $p=\Omega(\log(\kappa/\epsilon)/\log\log(\kappa/\epsilon))$ in Definition 2.2, we can run F^{2}SA-q with $q \asymp \log(\kappa/\epsilon)/\log\log(\kappa/\epsilon)$ and the $\mathcal{O}(q\kappa^9\epsilon^{-4}(\kappa/\epsilon)^{2/q}\log(\kappa/\epsilon))$ complexity in Theorem 3.1 simplifies to $\mathcal{O}(\kappa^9\epsilon^{-4}\log^3(\kappa/\epsilon)/\log\log(\kappa/\epsilon)) = \tilde{\mathcal{O}}(\kappa^9\epsilon^{-4})$, which matches the best-known complexity for HVP-based methods (Ji et al., 2021) under stochastic Hessian assumption (5).*

In the upcoming section, we will derive an $\Omega(\epsilon^{-4})$ lower bound to prove that the F^{2}SA-p is near-optimal in the above highly-smooth region if the condition number κ is a constant. We leave the study of optimal complexity for non-constant κ to future work.

Comparison of results for odd p and even p. Note that by Lemma 3.1 when p is odd, we need to use $p+1$ points to construct the estimator, which means the algorithm needs to solve $p+1$ lower-level problems in each iteration to achieve an $\mathcal{O}(\nu^p)$ error guarantee. In contrast, when p is even, p points are enough since the pth-order central difference estimator satisfies that $\alpha_0=0$. It suggests that even when p is odd, the algorithm designed for odd p may still be better. For instance, the F^{2}SA-2 may always be a better choice than F^{2}SA since its benefits *almost come for free*: (1) it still only needs to solve 2 lower-level problems as the F^{2}SA method, which means the per-iteration complexity remains the same. (2) Although the improved complexity of F^{2}SA-2 relies on the second-order smooth condition, without such a condition, its error guarantee in hyper-gradient estimation will only degenerate to a first-order one, which means it is at least as good as F^{2}SA.

4 An $\Omega(\epsilon^{-4})$ Lower Bound

In this section, we prove an $\Omega(\epsilon^{-4})$ lower bound for stochastic bilevel optimization via a reduction to single-level optimization. Our lower bound holds for any randomized algorithms $\mathtt{A}$, which can be defined as a sequence of measurable mappings $\{\mathtt{A}_t\}_{t=1}^T$ that is defined recursively by

$$(\boldsymbol{x}_{t+1},\boldsymbol{y}_{t+1}) = \mathtt{A}_t\left(r, F(\boldsymbol{x}_0,\boldsymbol{y}_0), G(\boldsymbol{x}_0,\boldsymbol{y}_0)), \cdots, F(\boldsymbol{x}_t,\boldsymbol{y}_t), G(\boldsymbol{x}_t,\boldsymbol{y}_t))\right), \quad t\in\mathbb{N}_+, \tag{11}$$

where r is a random seed drawn at the beginning to produce the queries, and F, G are the stochastic gradient estimators that satisfy Assumption 2.1. Without loss of generality, we assume that $(\boldsymbol{x}_0,\boldsymbol{y}_0)=(\mathbf{0},\mathbf{0})$. Otherwise, we can prove the same lower bound by shifting the functions.

The construction. We construct a separable bilevel instance such that the upper-level function $f(\boldsymbol{x},\boldsymbol{y}) \equiv f_{\boldsymbol{U}}(\boldsymbol{x})$ and its stochastic gradient align with the hard instance in (Arjevani et al., 2023), while the lower-level function is the simple quadratic $g(\boldsymbol{x},y)\equiv g(y)=\mu y^2/2$ with deterministic gradients. We defer the concrete construction to Appendix E. For this separable bilevel instance, we can show that for any randomized algorithm defined in Eq. (11) that uses oracles $(F_{\boldsymbol{U}}, G)$, the progress in $\boldsymbol{x}$ can be simulated by another randomized algorithm that only uses $F_{\boldsymbol{U}}$, meaning that the single-level lower bound (Arjevani et al., 2023) also holds.

Theorem 4.1 (Lower bound)**.** *There exist numerical constants $c>0$ such that for all $\Delta>0$, $L_1,\cdots,L_{p+1}>0$ and $\epsilon\le c\sqrt{L_1\Delta}$, there exists a distribution over the function class $\mathcal{F}^{nc\text{-}sc}(L_0,\cdots,L_{p+1},\mu,\Delta)$ and the stochastic gradient estimators satisfying Assumption 2.1, such that any randomized algorithm $\mathtt{A}$ defined as Eq. (11) can not find an ϵ-stationary point of $\varphi(\boldsymbol{x})=f(\boldsymbol{x},\boldsymbol{y}^*(\boldsymbol{x}))$ in less than $\Omega(\Delta L_1\sigma^2\epsilon^{-4})$ SFO calls.*

Below, we give a detailed discussion on the constructions in related works.

Comparison to other bilevel lower bounds. Dagréou et al. (2024) proved lower bounds for finite-sum bilevel optimization via a similar reduction to single-level optimization. However, the direct extension of their construction in the fully stochastic setting gives $f(\boldsymbol{x},\boldsymbol{y})=f_{\boldsymbol{U}}(\boldsymbol{y})$ and $g(\boldsymbol{x},\boldsymbol{y})=(\boldsymbol{x}-\boldsymbol{y})^2$, where the high-order derivatives of $f(\boldsymbol{x},\boldsymbol{y})$ not $\mathcal{O}(1)$-Lipschitz in $\boldsymbol{y}$ and thus violates our assumptions. Kwon et al. (2024a) also proved an $\Omega(\epsilon^{-4})$ lower bound for stochastic bilevel optimization. However, their construction $f(\boldsymbol{x}, y) = y$ and $g(\boldsymbol{x}, y) = (f_{\boldsymbol{U}}(\boldsymbol{x}) - y)^2$ violate the first-order smoothness of $g(\boldsymbol{x}, y)$ in $\boldsymbol{x}$ when y is far way from $f_{\boldsymbol{U}}(\boldsymbol{x})$. In this work, we use a fully separable construction to avoid all the aforementioned issues in other works.

5 EXPERIMENTS

In this section, we conduct numerical experiments to verify our theory. Following (Grazzi et al., 2020; Ji et al., 2021), we consider the "learn-to-regularize" problem of logistic regression (Example 2.2) on the "20 Newsgroup" dataset, which provably satisfies the highly smooth assumption of any order. The dataset contains 18,000 samples, each sample consists of a feature vector in dimensional $130,107$ vector and a label that takes value in $\{1, \cdots, 20\}$. We compare our proposed method $\text{F}^2\text{SA-}p$ with $p \in \{2, 3, 5, 8, 10\}$ with both the previous best fully first-order method F^2SA (Kwon et al., 2023; Chen et al., 2025b) and other Hessian-vector-product-based methods stocBiO (Ji et al., 2021), MRBO and VRBO (Yang et al., 2021). We also include a baseline "w/o Reg" that means the training result of SGD without tuning any regularization. For all the algorithms, we search the other hyperparameters (including η_x, η_y, ν) in a logarithmic scale with base 10. We run the algorithms with $K = 10$ iterations in the inner loop, and $T = 1000$ iterations in the outer loop, and report the test loss/accuracy *v.s.* the number of outer-loop iterations t in Figure 1. To demonstrate the potential of our methods on nonsmooth nonconvex problems, we also provide additional experiments on a 5-layer multilayer perceptron (MLP) network with ReLU activation in Appendix F.

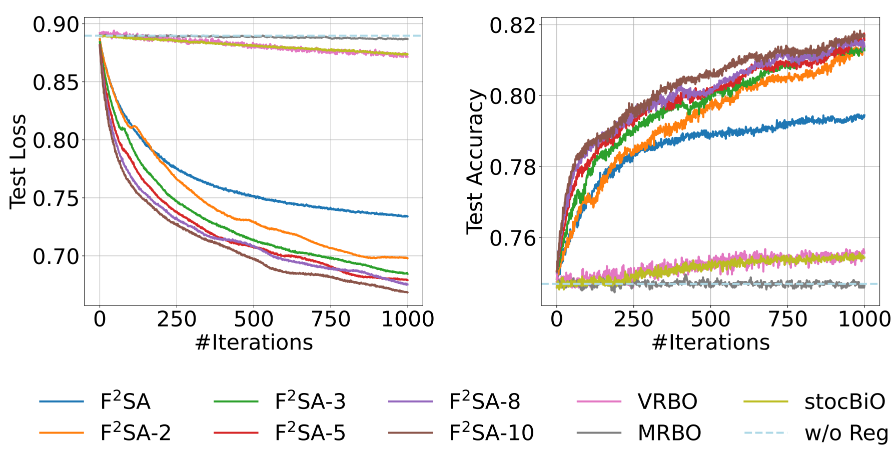

Figure 1: Performances of different algorithms on Example 2.2.

6 CONCLUSIONS AND FUTURE WORKS

This paper proposes a class of fully first-order method $\text{F}^2\text{SA-}p$ that achieves the $\tilde{\mathcal{O}}(p\epsilon^{-4-2/p})$ SFO complexity for pth-order smooth bilevel problems. Our result generalized the best-known $\tilde{\mathcal{O}}(\epsilon^{-6})$ result (Kwon et al., 2024a; Chen et al., 2025b) from $p = 1$ to any $p \in \mathbb{N}_+$. We also complement our result with an $\Omega(\epsilon^{-4})$ lower bound to show that our method is near-optimal when $p = \Omega(\log \epsilon^{-1} / \log \log \epsilon^{-1})$. Nevertheless, a gap still exists when p is small, and how to fill it even for the basic setting $p = 1$ is an open problem. Another open problem is whether our theory can be extended our theory to structured nonconvex-nonconvex bilevel problems studied by many recent works (Kwon et al., 2024b; Chen et al., 2024; 2025a; Jiang et al., 2025; Xiao et al., 2023; Xiao & Chen, 2025). In addition, it will also be interesting to further improve the convergence rate of our methods by combining them with variance-reduction (Fang et al., 2018; Cutkosky & Orabona, 2019) or momentum techniques (Fang et al., 2019; Cutkosky & Mehta, 2020).

REFERENCES

Zeyuan Allen-Zhu and Yuanzhi Li. Neon2: Finding local minima via first-order oracles. In *NeurIPS*, 2018.

Yossi Arjevani, Yair Carmon, John C. Duchi, Dylan J. Foster, Ayush Sekhari, and Karthik Sridharan. Second-order information in non-convex stochastic optimization: Power and limitations. In *COLT*, 2020.

Yossi Arjevani, Yair Carmon, John C. Duchi, Dylan J. Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, 199(1):165–214, 2023.

Kendall Atkinson and Weimin Han. Finite difference method. *Theoretical Numerical Analysis: A Functional Analysis Framework*, pp. 249–271, 2005.

Fan Bao, Guoqiang Wu, Chongxuan Li, Jun Zhu, and Bo Zhang. Stability and generalization of bilevel programming in hyperparameter optimization. In *NeurIPS*, 2021.

Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points i. *Mathematical Programming*, 184(1):71–120, 2020.

Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points ii: first-order methods. *Mathematical Programming*, 185(1):315–355, 2021.

El Mahdi Chayti and Martin Jaggi. A new first-order meta-learning algorithm with convergence guarantees. *arXiv preprint arXiv:2409.03682*, 2024.

He Chen, Jiajin Li, and Anthony Man-cho So. Set smoothness unlocks clarke hyper-stationarity in bilevel optimization. In *NeurIPS*, 2025a.

Lesi Chen, Jing Xu, and Jingzhao Zhang. On finding small hyper-gradients in bilevel optimization: Hardness results and improved analysis. In *COLT*, 2024.

Lesi Chen, Yaohua Ma, and Jingzhao Zhang. Near-optimal nonconvex-strongly-convex bilevel optimization with fully first-order oracles. *JMLR*, 2025b.

Ashok Cutkosky and Harsh Mehta. Momentum improves normalized SGD. In *ICML*, 2020.

Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex SGD. In *NeurIPS*, 2019.

Mathieu Dagréou, Thomas Moreau, Samuel Vaiter, and Pierre Ablin. A lower bound and a near-optimal algorithm for bilevel empirical risk minimization. In *AISTATS*, 2024.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *NeurIPS*, 2018.

Cong Fang, Zhouchen Lin, and Tong Zhang. Sharp analysis for nonconvex SGD escaping from saddle points. In *COLT*, 2019.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, 2018.

Ankit Garg, Robin Kothari, Praneeth Netrapalli, and Suhail Sherif. Near-optimal lower bounds for convex optimization for all orders of smoothness. In *NeurIPS*, 2021.

Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. In *ICML*, 2020.

Minhui Huang, Xuxing Chen, Kaiyi Ji, Shiqian Ma, and Lifeng Lai. Efficiently escaping saddle points in bilevel optimization. *JMLR*, 26(1):1–61, 2025.

Kaiyi Ji, Junjie Yang, and Yingbin Liang. Bilevel optimization: Convergence analysis and enhanced design. In *ICML*, 2021.

Liuyuan Jiang, Quan Xiao, Lisha Chen, and Tianyi Chen. Beyond value functions: Single-loop bilevel optimization under flatness conditions. In *NeurIPS*, 2025.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. In *ICML*, 2017.

Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M. Kakade, and Michael I. Jordan. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *Journal of the ACM*, 68(2):1–29, 2021.

Ishtiaq Rasool Khan, Ryoji Ohba, and Noriyuki Hozumi. Mathematical proof of closed form expressions for finite difference approximations based on taylor series. *Journal of Computational and Applied Mathematics*, 150(2):303–309, 2003.

Prashant Khanduri, Siliang Zeng, Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A near-optimal algorithm for stochastic bilevel optimization via double-momentum. In *NeurIPS*, 2021.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3): 455–500, 2009.

Guy Kornowski and Ohad Shamir. Oracle complexity in nonsmooth nonconvex optimization. *JMLR*, pp. 1–44, 2022.

Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, and Robert D. Nowak. A fully first-order method for stochastic bilevel optimization. In *ICML*, 2023.

Jeongyeol Kwon, Dohyun Kwon, and Hanbaek Lyu. On the complexity of first-order methods in stochastic bilevel optimization. In *ICML*, 2024a.

Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, and Robert D. Nowak. On penalty methods for nonconvex bilevel optimization and first-order stochastic approximation. In *ICLR*, 2024b.

Huan Li and Zhouchen Lin. Restarted nonconvex accelerated gradient descent: No more polylogarithmic factor in the in the $\mathcal{O}(\epsilon^{-7/4})$ complexity. *JMLR*, 2023.

Martin W Licht. Higher-order chain rules for tensor fields, generalized bell polynomials, and estimates in orlicz-sobolev-slobodeckij and total variation spaces. *Journal of Mathematical Analysis and Applications*, 534(1):128005, 2024.

Bo Liu, Mao Ye, Stephen Wright, Peter Stone, and Qiang Liu. Bome! bilevel optimization made easy: A simple first-order approach. In *NeurIPS*, 2022.

Risheng Liu, Xuan Liu, Shangzhi Zeng, Jin Zhang, and Yixuan Zhang. Value-function-based sequential minimization for bi-level optimization. *TPAMI*, 45(12):15930–15948, 2023.

Zhaosong Lu and Sanyou Mei. First-order penalty methods for bilevel optimization. *SIAM Journal on Optimization*, 34(2):1937–1969, 2024.

Luo Luo, Yujun Li, and Cheng Chen. Finding second-order stationary points in nonconvex-strongly-concave minimax optimization. In *NeurIPS*, 2022.

Matthew Mackay, Paul Vicol, Jonathan Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *ICLR*, 2019.

Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

Rui Pan, Jipeng Zhang, Xingyuan Pan, Renjie Pi, Xiaoyu Wang, and Tong Zhang. ScaleBiO: Scalable bilevel optimization for LLM data reweighting. *arXiv preprint arXiv:2406.19976*, 2024.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *ICML*, 2016.

Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *NeurIPS*, volume 32, 2019.

Han Shen and Tianyi Chen. On penalty-based bilevel gradient descent method. In *ICML*, 2023.

Han Shen, Pin-Yu Chen, Payel Das, and Tianyi Chen. Seal: Safety-enhanced aligned llm fine-tuning via bilevel data selection. In *ICLR*, 2025a.

Han Shen, Quan Xiao, and Tianyi Chen. On penalty-based bilevel gradient descent method. *Mathematical Programming*, pp. 1–51, 2025b.

Quan Xiao and Tianyi Chen. Unlocking global optimality in bilevel optimization: A pilot study. In *ICLR*, 2025.

Quan Xiao, Songtao Lu, and Tianyi Chen. An generalized alternating optimization method for bilevel problems under the polyak-łojasiewicz condition. In *NeurIPS*, 2023.

Yi Xu, Rong Jin, and Tianbao Yang. First-order stochastic algorithms for escaping from saddle points in almost linear time. In *NeurIPS*, 2018.

Haikuo Yang, Luo Luo, Chris Junchi Li, and Michael I Jordan. Accelerating inexact hypergradient descent for bilevel optimization. *arXiv preprint arXiv:2307.00126*, 2023a.

Junjie Yang, Kaiyi Ji, and Yingbin Liang. Provably faster algorithms for bilevel optimization. In *NeurIPS*, 2021.

Yifan Yang, Peiyao Xiao, and Kaiyi Ji. Achieving $\mathcal{O}(\epsilon^{-1.5})$ complexity in hessian/jacobian-free stochastic bilevel optimization. In *NeurIPS*, 2023b.

Jingzhao Zhang, Hongzhou Lin, Stefanie Jegelka, Suvrit Sra, and Ali Jadbabaie. Complexity of finding stationary points of nonconvex nonsmooth functions. In *ICML*, 2020.

A Notations for Tensors

We follow the notation of tensors used by Kolda & Bader (2009). For two p-way tensors $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ and $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$, their inner product $z = \langle \mathcal{X}, \mathcal{Y} \rangle$ is defined as

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_q=1}^{n_p} \mathcal{X}_{i_1, i_2, \cdots, i_p} \mathcal{Y}_{i_1, i_2, \cdots, i_p}.$$

For two tensors $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots n_p}$ and $\mathcal{Y} \in \mathbb{R}^{m_1 \times m_2 \cdots \times m_q}$, their outer product $\mathcal{Z} = \mathcal{X} \otimes \mathcal{Y}$ is a tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p \times m_1 \times m_2 \times \cdots \times m_q}$ whose elements are defined as

$$(\mathcal{X} \otimes \mathcal{Y})_{i_1, i_2, \cdots, i_p, j_1, j_2, \cdots, j_q} = \mathcal{X}_{i_1, i_2, \cdots, i_p} \mathcal{Y}_{i_1, i_2, \cdots, i_p}.$$

The operator norm of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is defined as

$$\|\mathcal{X}\| = \sup_{\|\boldsymbol{u}_i\|=1, i=1, \cdots, p} \langle \mathcal{X}, \boldsymbol{u}_1 \otimes \boldsymbol{u}_2 \otimes \cdots \otimes \boldsymbol{u}_p \rangle.$$

Equipped with the notion of norm, we say a mapping $\mathcal{T} : \mathbb{R} \to \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is D-bounded if

$$\|\mathcal{T}(\boldsymbol{x})\| \leq D, \quad \forall \boldsymbol{x} \in \mathbb{R}.$$

We say $\mathcal{T}$ is C-Lipschitz continuous if

$$\|\mathcal{T}(\boldsymbol{x}) - \mathcal{T}(\boldsymbol{y})\| \leq C\|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}.$$

B Proof of Lemma 3.1

Proof. If $\psi^{(p)}(\nu)$ is C-Lipschitz continuous in ν, then by Taylor's theorem we have

$$\psi(\nu) = \psi(0) + \sum_{k=1}^{p} \frac{(j\nu)^k}{k!}\psi^{(k)}(0) + \mathcal{O}\left(C\nu^{p+1}\right). \tag{12}$$

Now, we analyze the case when p is even or odd separately.

If p is even. The estimator we use is known as the pth-order central difference, whose coefficients are known (Khan et al., 2003). Let $n = p/2$. We select coefficients $\{\alpha_j\}_{j=-n}^{n}$ such that

$$\alpha_j = -\alpha_{-j}, \quad \forall j = 0, 1, \cdots, n.$$

Then, summing up Eq. (12) with coefficients α_j gives

$$\frac{1}{\nu}\sum_{j=-n}^{j=n} \alpha_j \psi(j\nu) = \underbrace{2\sum_{j=1}^{n} \alpha_j \sum_{k=1,3,\cdots}^{n-1} \frac{j^k \nu^{k-1}}{k!}\psi^{(k)}(0)}_{(*)} + \mathcal{O}\left(C\nu^{p}\right).$$

To let term (*) be equivalent to $\psi'(0)$, we let $\{\alpha_j\}_{j=1}^{n}$ satisfy the following equations:

$$2\sum_{j=1}^{n} \alpha_j j^k = \mathbf{1}_{k=1}, \quad \forall k = 1, 3, \cdots, n-1,$$

which is equivalent to let $\{j\alpha_j\}_{j=1}^{n}$ satisfy the following linear equation

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1^2 & 2^2 & 3^2 & \cdots & n^2 \\ 1^4 & 2^4 & 3^4 & \cdots & n^4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1^{2(n-1)} & 2^{2(n-1)} & 3^{2(n-1)} & \cdots & n^{2(n-1)} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ 2\alpha_2 \\ 3\alpha_3 \\ \vdots \\ n\alpha_n \end{pmatrix} = \begin{pmatrix} 1/2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Now we solve this linear equation to determine the values of $\{\alpha_j\}_{j=1}^{n}$. Let $\boldsymbol{A}$ be the coefficient matrix of this linear equation, and let $\boldsymbol{A}_j$ be the matrix such that the jth column of $\boldsymbol{A}$ is replaced by the standard unit vector $(1, 0, \cdots, 0)^\top$. By Cramer's rule, we have

$$2j\alpha_j = \frac{\det(\boldsymbol{A}_j)}{\det(\boldsymbol{A})}, \quad j = 1, \cdots, n.$$

By observation, we can find that both $\boldsymbol{A}$ and $\boldsymbol{A}_j$ are Vandermonde matrices. Therefore, we can explicitly calculate both $\det(\boldsymbol{A})$ and $\det(\boldsymbol{A}_j)$ according to the determinant formula of Vandermonde matrices, which leads to

$$2j\alpha_j = \frac{(-1)^{j-1}\cdot((j-1)!)^2 \cdot (n!)^2 \cdot j! \cdot (2j)!}{(j!)^2 \cdot (j-1)! \cdot (n-j)! \cdot (2j-1)! \cdot (n+j)!} = \frac{2(-1)^{j-1}(n!)^2}{(n-j)!\cdot(n+j)!}.$$

Therefore, we have

$$\alpha_j = \frac{(-1)^{j-1}(n!)^2}{j\cdot(n-j)!\cdot(n+j)!},$$

from which it is clear that $|\alpha_j| \le 1/j$.

If p is odd. Instead of using the known pth-order forward difference (Khan et al., 2003) for which we find that the coefficients will be exponentially large in p, we motivate from the pth-order central difference above to obtain a stable estimator by leveraging negative points. Let $n = (p+1)/2$.

We select coefficients $\{\alpha_j\}_{j=1-n}^{n}$ that satisfy the constraint $\sum_{j=1-n}^{n} \alpha_j = 0$. Then, summing up Eq. (12) with coefficients α_j gives

$$\frac{1}{\nu} \sum_{j=1-n}^{j=n} \alpha_j \psi(j\nu) = \underbrace{\sum_{j=1-n}^{j=n} \alpha_j \sum_{k=1}^{p} \frac{j^k \nu^{k-1}}{k!} \psi^{(k)}(0)}_{(*)} + \mathcal{O}\left(C\nu^p\right).$$

To let term (*) be equivalent to $\psi'(0)$, we let $\{\alpha_j\}_{j=1-n}^{n}$ satisfy the following equations:

$$\sum_{j=1-n}^{n} \alpha_j j^k = \mathbf{1}_{k=1}, \quad \forall k = 1, 2, \cdots, p,$$

which is equivalent to let $\{\hat{\alpha}_j\}_{j=(1-n), j\neq 0}^{n}$ satisfy the following linear equation

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & \cdots & 1 \\ 1-n & 2-n & 3-n & \cdots & -1 & 1 & \cdots & n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ (1-n)^{2n+1} & (2-n)^{2n+1} & (3-n)^{2n+1} & \cdots & (-1)^{2n+1} & 1^{2n+1} & \cdots & n^{2n+1} \end{pmatrix} \begin{pmatrix} \hat{\alpha}_{1-n} \\ \hat{\alpha}_{2-n} \\ \vdots \\ \hat{\alpha}_n \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

where we denote $\hat{\alpha}_j = j\alpha_j$ for $j = 1-n, \cdots, -1$ and $1, \cdots, n$. Now we solve this linear equation to determine the values of $\{\hat{\alpha}_j\}$. Let $\boldsymbol{A}$ be the coefficient matrix of this linear equation, and let $\boldsymbol{A}_j$ be the matrix such that the jth column of $\boldsymbol{A}$ is replaced by the standard unit vector $(1, 0, \cdots, 0)^\top$. By Cramer's rule, we have

$$\hat{\alpha}_j = \frac{\det(\boldsymbol{A}_j)}{\det(\boldsymbol{A})}, \quad j = 1, \cdots, n.$$

Similar to the case of even p, both $\boldsymbol{A}$ and $\boldsymbol{A}_j$ are Vandermonde matrices. Therefore, we can explicitly calculate both $\det(\boldsymbol{A})$ and $\det(\boldsymbol{A}_j)$ according to the determinant formula of Vandermonde matrices. Then, for $j = 1, \cdots, n$, we can obtain that

$$\alpha_j = \frac{\hat{\alpha}_j}{j} = \frac{(-1)^{j-1}(j-1)!(n-1)!n!j!}{j \cdot j!(j-1)!(n+j-1)!(n-j)!} = \frac{(-1)^{j-1}(n-1)!n!}{j(n+j-1)!(n-j)!}.$$

Similarly, for $j = 1, \cdots, n-1$, we can obtain that

$$\alpha_{-j} = \frac{\hat{\alpha}_{-j}}{-j} = \frac{(-1)^{j}(n-1)!(j-1)!n!j!}{j \cdot j!(n+j-1)!(j-1)!(n+j)!} = \frac{(-1)^{j}(n-1)!n!}{j(n-j-1)!(n+j)!}.$$

Therefore, it is easy to see that $|\alpha_j| \leq 1/j$ for $j = 1, \cdots, n$, and $|\alpha_{-j}| \leq 1/j$ for $j = 1, \cdots, n-1$. Finally, we calculate α_0 from the constraint $\sum_{j=1-n}^{n} \alpha_j = 0$, which leads to

$$\alpha_0 = -\underbrace{\sum_{j=1}^{n} \frac{(-1)^{j-1}(n-1)!n!}{j(n+j-1)!(n-j)!}}_{S_1} - \underbrace{\sum_{j=1}^{n-1} \frac{(-1)^{j}(n-1)!n!}{j(n-j-1)!(n+j)!}}_{S_2}. \tag{13}$$

We claim that $\alpha_0 = -1/n$ and hence $|\alpha_0| \leq 1$. We prove our claim by calculating the values of S_1 and S_2 to obtain α_0. For S_1, we have

$$S_1 = \sum_{j=1}^{n} (-1)^{j-1} \binom{n}{j} \frac{(n-1)!(j-1)!}{(n+j-1)!}.$$

The fraction on the right is the Beta function $B(j, n)$, which can be represented as the integral $B(j, n) = \int_0^1 x^{j-1}(1-x)^{n-1} \mathrm{d}x$. Therefore,

$$\begin{aligned} S_1 &= \sum_{j=1}^{n} (-1)^{j-1} \binom{n}{j} \int_0^1 x^{j-1}(1-x)^{n-1} \mathrm{d}x \\ &= \int_0^1 (1-x)^{n-1} \left(\sum_{j=1}^{n} (-1)^{j-1} \binom{n}{j} x^{j-1} \right) \mathrm{d}x \\ &= \int_0^1 \frac{(1-x)^{n-1}}{x} \left(\sum_{j=1}^{n} (-1)^{j-1} \binom{n}{j} x^{j} \right) \mathrm{d}x. \end{aligned}$$

Substituting the binomial expansion $(1-x)^n = 1 + \sum_{j=0}^{n} \binom{n}{j}(-x)^j$, we then have

$$S_1 = \int_0^1 \frac{(1-x)^{n-1}}{x}\left(1-(1-x)^n\right)\mathrm{d}x.$$

Let $y = 1-x$. We then have

$$S_1 = \int_0^1 \frac{y^{n-1}}{1-y}(1-y^n)\mathrm{d}y.$$

Substituting the geometric series sum $\frac{1-y^n}{1-y} = \sum_{k=0}^{n-1} y^k$, we then have

$$S_1 = \int_0^1 y^{n-1}\left(\sum_{k=0}^{n-1} y^k\right)\mathrm{d}y = \int_0^1 \sum_{k=0}^{n-1} y^{n+k-1}\mathrm{d}y = \sum_{k=0}^{n-1}\int_0^1 y^{n+k-1}\mathrm{d}y = \sum_{k=0}^{n-1}\frac{1}{n+k}.$$

Following similar steps, we can also obtain that

$$S_2 = -\sum_{k=0}^{n-2}\frac{1}{n+k+1}.$$

Now, for $\alpha_0 = -S_1 - S_2$, the summation terms cancel out perfectly, which leads to $\alpha_0 = -1/n$.

□

C PROOF OF LEMMA 3.2

The proof relies on the high-dimensional version of the Faà di Bruno formula. To formally state the result, we define the following notions. For a mapping $\mathcal{T} : \mathbb{R}^m \to \mathbb{R}^{n_1\times\cdots\times n_q}$, we define its kth-order directional derivative evaluated at $\boldsymbol{z} \in \mathbb{R}^m$ along the direction $(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_k)$ as

$$\nabla^k_{\boldsymbol{u}_1,\cdots,\boldsymbol{u}_k}\mathcal{T}_{|\boldsymbol{z}} = \nabla^k\mathcal{T}_{|\boldsymbol{z}}(\boldsymbol{u}_1,\cdots,\boldsymbol{u}_k).$$

We let the symmetric products of $\boldsymbol{u}_1,\cdots,\boldsymbol{u}_k$ as

$$\boldsymbol{u}_1 \vee \boldsymbol{u}_2 \vee \cdots \vee \boldsymbol{u}_k = \frac{1}{k!}\sum_{\pi\in\mathrm{Perm}(k)} \boldsymbol{u}_{\pi(1)} \otimes \boldsymbol{u}_{\pi(2)} \otimes \cdots \otimes \boldsymbol{u}_{\pi(k)},$$

where $\mathrm{Perm(k)}$ denotes the set of permutations of $\{1, 2, \cdots, k\}$. Also, we define the set of all (unordered) partitions of a set A into k pairwise disjoint non-empty sets as

$$\mathcal{P}(A,k) = \left\{\boldsymbol{P} = (P_1,\cdots,P_k) \subseteq \mathcal{B}(A) \mid A = \cup_{j=1}^k P_j;\ \emptyset \notin \boldsymbol{P};\ P_i \cap P_j = \emptyset, \forall i<j\right\},$$

where $\mathcal{B}(A)$ is the power set of A, *i.e.*, the set of all subsets of A. We also abbreviate $\mathcal{P}(\{1:q\},k)$ as $\mathcal{P}(q,k)$. Using the above notions, we have the following result.

Lemma C.1 ((Licht, 2024, Proposition 3.1))**.** *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two mappings. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are k-times differentiable at the point $\boldsymbol{z}$ and $\mathcal{T}_1(\boldsymbol{z})$, respectively, then the composite mapping $\mathcal{T}_2 \circ \mathcal{T}_1$ is k-times differentiable at the point $\boldsymbol{z}$ and we have*

$$\nabla^q(\mathcal{T}_2\circ\mathcal{T}_1)_{|\boldsymbol{z}}(\vee_{i=1}^q\boldsymbol{u}_i) = \sum_{\substack{1\le k\le q,\\ \boldsymbol{P}\in\mathcal{P}(q,k)}} \nabla^k\mathcal{T}_{2|\mathcal{T}_1(\boldsymbol{z})}\left(\nabla^{|P_1|}\mathcal{T}_{1|\boldsymbol{z}}(\vee_{i\in P_1}\boldsymbol{u}_i),\cdots\nabla^{|P_k|}\mathcal{T}_{1|\boldsymbol{z}}(\vee_{i\in P_k}\boldsymbol{u}_i)\right).$$

Recall Danskin's theorem that $\frac{\partial}{\partial\boldsymbol{x}}\ell_\nu(\boldsymbol{x}) = \frac{\partial}{\partial\boldsymbol{x}}g_\nu(\boldsymbol{x},\boldsymbol{y}^*_\nu(\boldsymbol{x}))$. We can apply Lemma C.1 with $\mathcal{T}_1 = \boldsymbol{y}^*_\nu(\boldsymbol{x})$ and $\mathcal{T}_1 = \frac{\partial}{\partial\boldsymbol{x}}g_\nu(\boldsymbol{x},\boldsymbol{y})$ to obtain that

$$\frac{\partial^{q+1}}{\partial\nu^q\partial\boldsymbol{x}}\ell_\nu(\boldsymbol{x}) = \sum_{\substack{1\le k\le q,\\ \boldsymbol{P}\in\mathcal{P}(q,k)}} \frac{\partial^{k+1}}{\partial\boldsymbol{y}^k\partial\boldsymbol{x}}g_\nu(\boldsymbol{x},\boldsymbol{y}^*_\nu(\boldsymbol{x}))\left(\frac{\partial^{|P_1|}}{\partial\nu^{|P_1|}}\boldsymbol{y}^*_\nu(\boldsymbol{x}),\cdots,\frac{\partial^{|P_k|}}{\partial\nu^{|P_k|}}\boldsymbol{y}^*_\nu(\boldsymbol{x})\right). \tag{14}$$

Symmetrically, using the first-order optimality condition $\frac{\partial}{\partial \boldsymbol{y}} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})) = 0$ and where the first identity uses the Lemma C.1 with $\mathcal{T}_1 = \boldsymbol{y}_\nu^*(\boldsymbol{x})$ and $\mathcal{T}_1 = \frac{\partial}{\partial \boldsymbol{y}} g_\nu(\boldsymbol{x}, \boldsymbol{y})$ yields that

$$0 = \sum_{\substack{1 \le k \le q, \\ \boldsymbol{P} \in \mathcal{P}(q,k)}} \frac{\partial^{k+1}}{\partial \boldsymbol{y}^{k+1}} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})) \left(\frac{\partial^{|P_1|}}{\partial \nu^{|P_1|}} \boldsymbol{y}_\nu^*(\boldsymbol{x}), \cdots, \frac{\partial^{|P_k|}}{\partial \nu^{|P_k|}} \boldsymbol{y}_\nu^*(\boldsymbol{x}) \right). \tag{15}$$

Since $\mathcal{P}(q,1)$ contains only one element, the above identity implies that

$$\begin{aligned} \frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x}) &= - \left(\nabla_{yy}^2 g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})) \right)^{-1} \sum_{\substack{2 \le k \le q, \\ \boldsymbol{P} \in \mathcal{P}(q,k)}} \boldsymbol{w}_{k,\boldsymbol{P}}, \\ \text{where } \ \boldsymbol{w}_{k,\boldsymbol{P}} &= \frac{\partial^{k+1}}{\partial \boldsymbol{y}^{k+1}} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})) \left(\frac{\partial^{|P_1|}}{\partial \nu^{|P_1|}} \boldsymbol{y}_\nu^*(\boldsymbol{x}), \cdots, \frac{\partial^{|P_k|}}{\partial \nu^{|P_k|}} \boldsymbol{y}_\nu^*(\boldsymbol{x}) \right). \end{aligned} \tag{16}$$

Based on Eq. (16), we can prove by induction that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν for all $q = 0, \cdots, p$. The induction base for $q = 0, 1$ is already proved by Chen et al. (2025b).

Lemma C.2 (Chen et al. (2025b, Lemma B.2 and B.5))**.** *Let $\nu \in (0, 1/(2\kappa)]$. Under Assumption 2.3 and 2.4, $\boldsymbol{y}_\nu^*(\boldsymbol{x})$ and $\frac{\partial}{\partial \nu} \boldsymbol{y}_\nu^*(\boldsymbol{x})$ is $\mathcal{O}(\kappa)$- and $\mathcal{O}(\kappa^3)$-Lipschitz continuous in ν, respectively.*

Since Eq. (16) also involves $(\nabla_{yy}^2 g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})))^{-1}$, we also need the following lemma that gives its boundedness and Lipschitz continuity constants.

Lemma C.3 (Chen et al. (2025b, Lemma B.1 and Eq. 18))**.** *Let $\nu \in (0, 1/(2\kappa)]$. Under Assumption 2.3 and 2.4, $(\nabla_{yy} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})))^{-1}$ is $2/\mu$-bounded and $\mathcal{O}(\kappa^2/\mu)$-Lipschitz continuous in ν.*

In the remaining proofs, we will use Eq. (16) prove by induction that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν, then we can easily use Eq. (14) to show that $\frac{\partial^{q+1}}{\partial \nu^q \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1}\bar{L})$-Lipschitz continuous in ν for all $q = 0, \cdots, p$. Note that the computational graph of either $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x}))$ or $\frac{\partial^{q+1}}{\partial \nu^q \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})$ in Eq. (14) or (16) defines a tree, where the root is output, the leaves are inputs, and the other nodes are the intermediate results in the computation. We can analyze the Lipschitz continuities of all the nodes from bottom to top using the following lemma.

Lemma C.4 (Luo et al. (2022, Lemma 12))**.** *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two tensor-to-tensor mappings. If $\mathcal{T}_1$ is D_1-bounded and C_1-Lipschitz continuous, $\mathcal{T}_2$ is D_2-bounded and C_2-Lipschitz continuous, then the product mapping $\mathcal{T}_1 \times \mathcal{T}_2$ is $D_1 D_2$-bounded and $(C_1 D_2 + C_2 D_1)$-Lipschitz continuous.*

Proof of Lemma 3.2. Now, we formally begin to prove by induction that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν for all $q = 0, \cdots, p$. Recall that the induction base follows Lemma C.2. In the following, we use the induction hypothesis that $\frac{\partial^k}{\partial \nu^k} \boldsymbol{y}_\nu^*(\boldsymbol{x}))$ is $\mathcal{O}(\kappa^{2k+1})$-Lipschitz continuous in ν for all $k = 0, \cdots, q-1$ to prove that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x}))$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν. We know that $\frac{\partial^{k+1}}{\partial \boldsymbol{y}^{k+1}} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x}))$ is $\mathcal{O}(\bar{L})$-bounded and $\mathcal{O}(\kappa \bar{L})$-Lipschitz continuous in ν. Therefore, we can use Lemma C.4 to conclude that each $\boldsymbol{w}_{k,\boldsymbol{P}}$ is $\mathcal{O}(\kappa^{\sum_{j=1}^k (2|P_j|-1)} \bar{L}) = \mathcal{O}(\kappa^{2q-k}\bar{L})$-bounded and $\mathcal{O}(\bar{L} \cdot \kappa^{2q-k+2} + \kappa \bar{L} \cdot \kappa^{2q-k}) = \mathcal{O}(\kappa^{2q-k+2}\bar{L})$-Lipschitz continuous in ν. It further implies that the summation $\boldsymbol{w} := \sum_{2 \le k \le q, \boldsymbol{P} \in \mathcal{P}(q,k)} \boldsymbol{w}_{k,\boldsymbol{P}}$ is $\mathcal{O}(\kappa^{2q-2}\bar{L})$-bounded and $\mathcal{O}(\kappa^{2q}\bar{L})$-Lipschitz continuous in ν. Then, we can recall Lemma C.3 that $(\nabla_{yy} g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})))^{-1}$ is $2/\mu$-bounded and $\mathcal{O}(\kappa^2/\mu)$-Lipschitz continuous in ν, and use Eq. (16) to finish the induction that $\frac{\partial^q}{\partial \nu^q} \boldsymbol{y}_\nu^*(\boldsymbol{x}) = - \left(\nabla_{yy}^2 g_\nu(\boldsymbol{x}, \boldsymbol{y}_\nu^*(\boldsymbol{x})) \right)^{-1} \boldsymbol{w}$ is $\mathcal{O}(\kappa^{2q+1})$-Lipschitz continuous in ν for all $q = 0, \cdots, p$. Finally, by analogy with the similarity of Eq. (14) and (16), we can follow the same analysis to show that $\frac{\partial^{q+1}}{\partial \nu^q \partial \boldsymbol{x}} \ell_\nu(\boldsymbol{x})$ is $\mathcal{O}(\kappa^{2q+1}\bar{L})$-Lipschitz continuous in ν for all $= 0, \cdots, p$. □

D PROOF OF THEOREM 3.1

In the main text, we only present the algorithm when p is even. The algorithm when p is odd follows a similar design, which is presented in Algorithm 2 for completeness. Our algorithms consist of a double loop, where the outer loop performs normalized SGD (NSGD) and the inner loop performs SGD. Before we give the formal proof, we first recall the convergence result for (N)SGD.

Algorithm 2 $\text{F}^2\text{SA-}p\ (\boldsymbol{x}_0, \boldsymbol{y}_0)$, odd p

1: $\boldsymbol{y}_0^j = \boldsymbol{y}_0,\ \forall j \in \mathbb{N}$
2: **for** $t = 0, 1, \cdots, T-1$
3: **parallel for** $j = -(p-1)/2, \cdots, (p+1)/2$
4: $\boldsymbol{y}_t^{j,0} = \boldsymbol{y}_t^j$
5: **for** $k = 0, 1, \cdots, K-1$
6: Sample random i.i.d indexes $\{(\xi_j^y, \zeta_j^y)\}$.
7: $\boldsymbol{y}_t^{j,k+1} = \boldsymbol{y}_t^{j,k} - \eta_y \left(j\nu F_y(\boldsymbol{x}_t, \boldsymbol{y}_t^{j,k}; \xi_j^y) + G_y(\boldsymbol{x}_t, \boldsymbol{y}_t^{j,k}; \zeta_j^y) \right)$
8: **end for**
9: $\boldsymbol{y}_{t+1}^j = \boldsymbol{y}_t^{j,K}$
10: **end parallel for**
11: Sample random i.i.d indexes $\{(\xi_i^x, \zeta_i^x)\}_{i=1}^S$.
12: Let $\{\alpha_j\}_{j=-(p-1)/2}^{(p+1)/2}$ be the pth-order finite difference coefficients defined in Lemma 3.1.
13: $\Phi_t = \frac{1}{S}\sum_{i=1}^S \sum_{j=-(p-1)/2}^{(p+1)/2} \alpha_j \left(jF_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j; \xi_i^x) + \dfrac{G_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j; \zeta_i^x)}{\nu} \right)$
14: $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_x \Phi_t / \|\Phi_t\|$
15: **end for**

Lemma D.1 (Cutkosky & Mehta (2020, Lemma 2))**.** *Consider the NSGD update $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta F_t/\|F_t\|$ to optimize a function $f: \mathbb{R}^d \to \mathbb{R}$ with L-Lipschitz continuous gradients. We have*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\boldsymbol{x}_t))\| \le \frac{3(f(\boldsymbol{x}_0) - \inf_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}))}{\eta T} + \frac{3L\eta}{2} + \frac{8}{T}\sum_{t=0}^{T-1}\mathbb{E}\|F_t - \nabla f(\boldsymbol{x}_t)\|.$$

Lemma D.2 (Kwon et al. (2024a, Lemma C.1))**.** *Consider the SGD update $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta F_t$ to optimize a μ-strongly convex function $f: \mathbb{R}^d \to \mathbb{R}$ with L-Lipschitz continuous gradients. Let $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x})$ be the unique minimizer to f. Suppose F_t is an unbiased estimator to $\nabla f(\boldsymbol{x}_t)$ with variance bounded by σ^2. Setting $\eta < 2/(\mu + L)$, we have*

$$\mathbb{E}\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2 \le (1-\mu\eta)^t \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 + \frac{\eta\sigma^2}{\mu}.$$

The following two lemmas are also useful in the analysis.

Lemma D.3 (Chen et al. (2025b, Lemma 4.1))**.** *Under Assumption 2.3, and 2.4, the hyper-objective $\varphi(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x}))$ is differentiable and has $L_\varphi = \mathcal{O}(\bar{L}\kappa^3)$-Lipschitz continuous gradients.*

Lemma D.4 (Chen et al. (2025b, Lemma B.6))**.** *Let $\nu \in (-1/\kappa, 1/\kappa)$. Under Assumption 2.3, and 2.4, the optimal (perturbed) lower-level solution mapping $\boldsymbol{y}_\nu^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}\in\mathbb{R}^{d_y}} \ell_v(\boldsymbol{x}, \boldsymbol{y})$ is 4κ-Lipschitz continuous in $\boldsymbol{x}$.*

Now we are ready to prove Theorem 3.1.

Proof of Theorem 3.1. We separately consider the complexity for the outer loop and the inner loop.

Outer Loop. According to Lemma D.3, the hyper-objective $\varphi(\boldsymbol{x})$ has $L_\varphi = \mathcal{O}(\bar{L}\kappa^3)$-Lipschitz continuous gradients. If we can guarantee the condition

$$\mathbb{E}\|\Phi_t - \nabla\varphi(\boldsymbol{x}_t)\| \le \frac{\epsilon}{32}, \quad t = 0, \cdots, T-1, \tag{17}$$

then we can further set $\eta_x = {}^{\epsilon}/_{6L_\varphi}$ and apply Lemma D.1 to conclude that the algorithm can provably find an ϵ-stationary point of $\varphi(\boldsymbol{x})$ in $T = \lceil {}^{6\Delta}/_{\epsilon\eta_x} \rceil = \mathcal{O}(\Delta L_1 \kappa^3 \epsilon^{-2})$ outer iterations.

Inner Loop. From the above analysis, the remaining goal is to show that the inner loop always returns Φ_t satisfying Eq. (17), which requires $\mathbb{E}\|\Phi_t - \nabla\varphi(\boldsymbol{x}_t)\| = \mathcal{O}(\epsilon)$ for all $t = 0, \cdots, T-1$. Note that the setting of mini-batch size $S = \Omega\left(\sigma^2/\nu^2\epsilon^2\right)$ ensures that

$$\begin{cases} \mathbb{E}\left\|\Phi_t - \sum_{j=-p/2}^{p/2} \alpha_j \left(j\nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j) + \dfrac{\nabla_x g(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j)}{\nu}\right)\right\| = \mathcal{O}(\epsilon), & p \text{ is even;} \\ \mathbb{E}\left\|\Phi_t - \sum_{j=-(p-1)/2}^{(p+1)/2} \alpha_j \left(j\nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j) + \dfrac{\nabla_x g(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}^j)}{\nu}\right)\right\| = \mathcal{O}(\epsilon), & p \text{ is odd.} \end{cases}$$

By Lemma 3.2 and Lemma 3.1, setting $\nu = \mathcal{O}((\epsilon/\bar{L}\kappa^{2p+1})^{1/p})$ can ensure that

$$\begin{cases} \left\|\nabla\varphi(\boldsymbol{x}_t) - \sum_{j=-p/2}^{p/2} \alpha_j \left(j\nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)) + \dfrac{\nabla_x g(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t))}{\nu}\right)\right\| = \mathcal{O}(\epsilon), & p \text{ is even;} \\ \left\|\nabla\varphi(\boldsymbol{x}_t) - \sum_{j=-(p-1)/2}^{(p+1)/2} \alpha_j \left(j\nabla_x f(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)) + \dfrac{\nabla_x g(\boldsymbol{x}_t, \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t))}{\nu}\right)\right\| = \mathcal{O}(\epsilon), & p \text{ is odd.} \end{cases}$$

Therefore, a sufficient condition of $\mathbb{E}\|\Phi_t - \nabla\varphi(\boldsymbol{x}_t)\| = \mathcal{O}(\epsilon)$ is

$$\begin{cases} \|\boldsymbol{y}_{t+1}^j - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| = \mathcal{O}(\nu\epsilon/L_1), \ \ \forall j = -p/2, \cdots, p/2, & p \text{ is even;} \\ \|\boldsymbol{y}_{t+1}^j - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| = \mathcal{O}(\nu\epsilon/L_1), \ \ \forall j = -(p-1)/2, \cdots, (p+1)/2, & p \text{ is odd.} \end{cases} \tag{18}$$

Our next goal is to show that our parameter setting fulfills Eq. (18). Note that for $\nu = \mathcal{O}(1/\kappa)$, the (perturbed) lower-level problem $g_{j\nu}(\boldsymbol{x}, \boldsymbol{y})$ is $\Omega(\mu)$-strongly convex in $\boldsymbol{y}$ and has $\mathcal{O}(L_1)$-Lipschitz continuous gradients jointly in $(\boldsymbol{x}, \boldsymbol{y})$. Therefore, if we set $\eta_y \lesssim 1/L_1$, then we can apply Lemma D.2 on the lower-level problem $g_{j\nu}(\boldsymbol{x}, \boldsymbol{y})$ to conclude that for ant j, we have

$$\mathbb{E}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\|^2 \le (1 - \mu\eta_y)^K \|\boldsymbol{y}_t - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\|^2 + \mathcal{O}(\eta_y\sigma^2/\mu).$$

Comparing it with Eq. (18), we can set $\eta_y = \mathcal{O}(\nu^2\epsilon^2/L_1\kappa\sigma^2)$ to ensure that for ant j, we have

$$\mathbb{E}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| \le (1 - \mu\eta_y)^K \|\boldsymbol{y}_t - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| + \mathcal{O}(\nu\epsilon/L_1).$$

Further, we can use Lemma D.4 and the triangle inequality to obtain that for ant j, we have

$$\mathbb{E}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| \le (1 - \mu\eta_y)^K (\|\boldsymbol{y}_t - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_{t-1})\| + 4\kappa\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\|) + \mathcal{O}(\nu\epsilon/L_1). \tag{19}$$

The recursion (19) implies our setting of K can ensure that Eq. (18) holds for all $t = 0, \cdots, T-1$. We give an induction-based proof. To let the induction base holds for $t = 1$, it suffices to set $K = \Omega(\log(RL_1/\nu\epsilon)/\mu\eta_y) = \Omega(\log(RL_1/\nu\epsilon)\kappa^2\sigma^2/\nu^2\epsilon^2)$, where $\|\boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_0) - \boldsymbol{y}^*(\boldsymbol{x}_0)\|^2 = \mathcal{O}(R)$ is due to the setting of $\nu = \mathcal{O}(R/\kappa)$ and the fact that $\boldsymbol{y}_\nu^*(\boldsymbol{x})$ is κ-Lipschitz in ν by Lemma C.2. Next, assume that we have already guaranteed Eq. (18) holds for iteration t, we prove that our setting of K implies Eq. (18) holds for iteration $t+1$. Note that the NSGD update in $\boldsymbol{x}$ means that $\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\| = \eta_x = \mathcal{O}(\epsilon/6L_1\kappa^3)$. Therefore, Eq. (19) in conjunction with the induction hypothesis indicates that

$$\mathbb{E}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_{j\nu}^*(\boldsymbol{x}_t)\| \lesssim (1 - \mu\eta_y)^K \left(\frac{\nu\epsilon}{L_1} + \frac{\epsilon}{L_1\kappa^2}\right) + \frac{\nu\epsilon}{L_1}.$$

Therefore, we know that to let Eq. (18) holds for iteration $t+1$, it suffices to let $K = \Omega(\log(1/\nu\kappa^2)/\mu\eta_y) = \Omega(\log(1/\nu\kappa^2)\kappa^2\sigma^2/\nu^2\epsilon^2)$. This finishes the induction.

Total Complexity. According to the above analysis, we set $\nu \asymp (\epsilon/\bar{L}\kappa^{2p+1})^{1/p}$, $S \asymp \sigma^2/\nu^2\epsilon^2$, $T \asymp \Delta L_1\kappa^3\epsilon^{-2}$, and $K \asymp \log(RL_1\kappa/\nu\epsilon)\kappa^2\sigma^2/\nu^2\epsilon^2$ to ensure that the algorithm provably find an ϵ-stationary point of $\varphi(\boldsymbol{x})$. Since $S \lesssim K$, the total complexity of the algorithm is

$$\begin{aligned} pT(S+K) &= \mathcal{O}(pTK) = \mathcal{O}\left(p \cdot \frac{\Delta L_1\kappa^3}{\epsilon^2} \cdot \frac{\kappa^2\sigma^2}{\nu^2\epsilon^2} \log\left(\frac{RL_1\kappa}{\nu\epsilon}\right)\right) \\ &= \mathcal{O}\left(\frac{p\Delta L_1\bar{L}^{2/p}\sigma^2\kappa^{9+2/p}}{\epsilon^{4+2/p}} \log\left(\frac{RL_1\kappa}{\nu\epsilon}\right)\right). \end{aligned}$$

$\square$

E Proof of Theorem 4.1

We prove our lower bound for stochastic nonconvex-strongly-convex bilevel optimization via a reduction to the lower bound for stochastic single-level nonconvex optimization (Arjevani et al., 2023). To state their lower bound, we first need to introduce the function class, oracle class, algorithm class, and the complexity measures.

Definition E.1. *Given any $\Delta > 0$ and $L_1 > 0$, we use $\mathcal{F}^{\mathrm{nc}}(L_1, \Delta)$ to denote the set of all smooth functions $f : \mathbb{R}^d \to \mathbb{R}$ that satisfies*

1. $f(\mathbf{0}) - \inf_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x}) \le \Delta$*;*
2. $\nabla f(\boldsymbol{x})$ *is L_1-Lipschitz continuous.*

Definition E.2. *Given a function $\mathbb{R}^d \to \mathbb{R}$, we use $\mathbb{O}(\sigma^2)$ to denote the set of all stochastic first-order oracles that return an unbiased stochastic estimator to ∇f with variance bounded by σ^2.*

Definition E.3. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function and $F : \mathbb{R}^d \to \mathbb{R}$ be the stochastic estimator to ∇f. A randomized first-order algorithm $\mathtt{A}$ consists of a distribution $\mathcal{P}_r$ over a measurable set $\mathcal{R}$ and a sequence of measurable mappings $\{\mathtt{A}_t\}_{t \in \mathbb{N}}$ such that*

$$\boldsymbol{x}_{t+1} = \mathtt{A}_t(r, F(\boldsymbol{x}_0), \cdots, F(\boldsymbol{x}_t)), \quad t \in \mathbb{N}_+,$$

where $r \sim \mathcal{P}_r$ is drawn a single time at the beginning of the protocol. We let $\mathcal{A}_{\mathrm{rand}}$ to denote the class of all the algorithms that follow the above protocol.

Definition E.4. *We define distributional complexity of $\mathcal{A}_{\mathrm{rand}}$ to find an ϵ-stationary point of the functions in $\mathcal{F}^{\mathrm{nc}}(L_1, \Delta)$ with oracle $\mathbb{O}(\sigma^2)$ as*

$$\mathrm{Compl}_\epsilon(L_1, \Delta, \sigma^2) = \sup_{\mathtt{O} \in \mathbb{O}(\sigma^2)} \sup_{\mathcal{P}_f \in \mathcal{P}[\mathcal{F}(\Delta, f)]} \inf_{\mathtt{A} \in \mathcal{A}_{\mathrm{rand}}} \inf\{t \in \mathbb{N} \mid \mathbb{E}\|\nabla f(\boldsymbol{x}_t)\| \le \epsilon\},$$

where the expectation is taken over the sampling of f from $\mathcal{P}_f$, the randomness in the oracle $\mathtt{O}$, and the randomness in the algorithm $\mathtt{A}$, $\{\boldsymbol{x}_t\}_{t \in \mathbb{N}}$ is the sequence generated by $\mathtt{A}$ running on f with oracle $\mathtt{O}$, and $\mathcal{P}[\mathcal{F}^{\mathrm{nc}}(L_1, \Delta)]$ denotes the set of all distributions over $\mathcal{F}^{\mathrm{nc}}(L_1, \Delta)$.

All the above definitions are merely restatements of (Arjevani et al., 2023, Section 2). Although Definition E.4 uses the definition of distributional complexity, by Yao's minimax principle is also a lower bound for the worst-case complexity. Now, we recall the construction in (Arjevani et al., 2023) for proving the $\Omega(\epsilon^{-4})$ lower bound. Formally, we define the randomized function

$$f_{\boldsymbol{U}}(\boldsymbol{x}) = \frac{L_1 \beta^2}{\bar{L}_1} f^{\mathrm{nc}}(\rho(\boldsymbol{U}^\top \boldsymbol{x}/\beta)) + \frac{L_1 \lambda}{2\bar{L}_1} \|\boldsymbol{x}\|^2, \tag{20}$$

where $\bar{L}_1 = 155$, $\beta = 4\bar{L}_1 \epsilon / L_1$, $\rho : \mathbb{R}^T \to \mathbb{R}^T$ is $\rho(\boldsymbol{x}) = \boldsymbol{x} \big/ \sqrt{1 + \|\boldsymbol{x}\|^2 / R^2}$, $R = 230\sqrt{T}$, $\lambda = 1/5$, and $f_T : \mathbb{R}^T \to \mathbb{R}$ is the nonconvex hard instance introduced by Carmon et al. (2020):

$$f^{\mathrm{nc}}(x) := -\Psi(1)\Psi(x_1) + \sum_{i=2}^{T} [\Phi(-x_{i-1})\Phi(-x_i) - \Psi(x_{i-1})\Phi(x_i)].$$

In the above, the component functions $\Psi, \Phi : \mathbb{R} \to \mathbb{R}$ are defined as

$$\Psi(t) = \begin{cases} 0, & t \le 1/2, \\ \exp(1 - 1/(2t-1)^2), & t < 1/2 \end{cases} \quad \text{and} \quad \Phi(t) = \sqrt{\mathrm{e}} \int_{-\infty}^{t} \exp(-t^2/2) \mathrm{d}t.$$

For the hard instance in Eq. (20), Arjevani et al. (2023) further defined the stochastic gradient estimator $F_{\boldsymbol{U}}$ as

$$F_{\boldsymbol{U}}(\boldsymbol{x}) = \frac{L_1}{\bar{L}_1} \left(\beta (\nabla \rho(\boldsymbol{x}))^\top \boldsymbol{U} F_T(\boldsymbol{U}^\top \rho(\boldsymbol{x})) + \lambda \boldsymbol{x} \right). \tag{21}$$

In the above, $F_T : \mathbb{R}^T \to \mathbb{R}^T$ is the stochastic gradient estimator of ∇f^{nc} defined by

$$[F_T(\boldsymbol{x})]_i = \nabla_i f^{\mathrm{nc}}(\boldsymbol{x}) \left(1 + \mathbf{1}_{i > \mathrm{prog}_{1/4}(x)} (\xi/\gamma - 1) \right), \quad \xi \sim \mathrm{Bernoulli}(\gamma),$$

where $\mathrm{prog}_\alpha(x) = \max\{i \ge 0 \mid |x_i| > \alpha\}$ and $\gamma = \min\{(46\epsilon)^2/\sigma^2, 1\}$. For the above construction, Arjevani et al. (2023) showed the following lower bound.

Theorem E.1 ((Arjevani et al., 2023, Theorem 3))**.** *There exist numerical constants $c, c' > 0$ such that for all $\Delta > 0$, $L_1 > 0$ and $\epsilon \le c\sqrt{L_1 \Delta}$, the construction of function $f_{\boldsymbol{U}} : \mathbb{R}^d \to \mathbb{R}$ and stochastic first-order oracle $F_{\boldsymbol{U}} : \mathbb{R}^d \to \mathbb{R}$ in Eq. (20) and (21) together give a distribution over the function class $\mathcal{F}^{\rm nc}(L_1, \Delta)$ and a stochastic first-order oracle $\mathtt{O} \in \mathbb{O}(\sigma^2)$ such that*

$$\mathrm{Compl}_\epsilon(L_1, \Delta, \sigma^2) \ge c' \Delta L_1 \sigma^2 \epsilon^{-4}.$$

Proof of Theorem 4.1. For any randomized algorithm $\mathtt{A}$ defined as Eq. (11) running it on our hard instance, we show that it can be simulated by another randomized algorithm running on the variable $\boldsymbol{x}$ such that Theorem E.1 can be applied. Since $G(y) = \mu y$ is a deterministic mapping we know that any randomized algorithm $\mathtt{A}$ induces a sequence of measurable mappings $\{\mathtt{A}'_t\}_{t \in \mathbb{N}}$ such that

$$(\boldsymbol{x}_t, y_t) = \mathtt{A}'_t(\xi, F(\boldsymbol{x}_0), \cdots, F(\boldsymbol{x}_{t-1}), y_0, \cdots, y_{t-1}).$$

Expanding the recursion for y_t shows that the above equation induces another sequence of measurable mappings $\{\mathtt{A}''_t\}_{t \in \mathbb{N}}$ such that

$$(\boldsymbol{x}_t, y_t) = \mathtt{A}''_t(\xi, F(\boldsymbol{x}_0), \cdots, F(\boldsymbol{x}_{t-1})).$$

Therefore, we can apply Theorem E.1 to complete the proof. □

F THE F^{2}SA-2 ALGORITHM

We present the realization of F^{2}SA-p when $p = 2$ in Algorithm 3 to further compare its procedure with the original F^{2}SA algorithm. Let $\lambda = 1/\nu$. We can observe that F^{2}SA (Kwon et al., 2023; Chen et al., 2025b) solves the following *asymmetric* penalty problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^{d_x}, \boldsymbol{y} \in \mathbb{R}^{d_y}} f(\boldsymbol{x}, \boldsymbol{y}) + \lambda \left(g(\boldsymbol{x}, \boldsymbol{y}) - \min_{\boldsymbol{z} \in \mathbb{R}^{d_y}} g(\boldsymbol{x}, \boldsymbol{z}) \right),$$

while F^{2}SA-2 solved the following *symmetric* penalty problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^{d_x}, \boldsymbol{y} \in \mathbb{R}^{d_y}} \frac{1}{2} \left(f(\boldsymbol{x}, \boldsymbol{y}) + \lambda f(\boldsymbol{x}, \boldsymbol{y}) - \min_{\boldsymbol{z} \in \mathbb{R}^{d_y}} (-f(\boldsymbol{x}, \boldsymbol{z}) + \lambda g(\boldsymbol{x}, \boldsymbol{z})) \right).$$

The latter is better since the symmetric form makes the first-order approximation error to $\nabla \varphi(\boldsymbol{x})$ perfectly cancel out and leave only the second-order error term. Therefore, in terms of the theoretical guarantee by Theorem 3.1, the $\tilde{\mathcal{O}}(\epsilon^{-5})$ upper bound of F^{2}SA-2 can improve the $\tilde{\mathcal{O}}(\epsilon^{-6})$ upper bound of F^{2}SA by a factor of ϵ^{-1}.

Algorithm 3 F^{2}SA-2 $(\boldsymbol{x}_0, \boldsymbol{y}_0)$

$\boldsymbol{z}_0 = \boldsymbol{y}_0$
for $t = 0, 1, \cdots, T-1$
    $\boldsymbol{y}_t^0 = \boldsymbol{y}_t,\ \boldsymbol{z}_t^0 = \boldsymbol{z}_t$
    for $k = 0, 1, \cdots, K-1$
        Sample random i.i.d indexes (ξ^y, ζ^y) and (ξ^z, ζ^z).
        $\boldsymbol{y}_t^{k+1} = \boldsymbol{y}_t^k - \eta_y \left(\nu F_y(\boldsymbol{x}_t, \boldsymbol{y}_t^k; \xi^y) + G_y(\boldsymbol{x}_t, \boldsymbol{y}_t^k; \zeta^y) \right)$
        $\boldsymbol{z}_t^{k+1} = \boldsymbol{z}_t^k - \eta_y \left(-\nu F_y(\boldsymbol{x}_t, \boldsymbol{z}_t^t; \xi^z) + G_y(\boldsymbol{x}_t, \boldsymbol{z}_t^k; \zeta^z) \right)$
    end for
    $\boldsymbol{y}_{t+1} = \boldsymbol{y}_t^K,\ \boldsymbol{z}_{t+1} = \boldsymbol{z}_t^K$
    Sample random i.i.d indexes $\{(\xi_i^x, \zeta_i^x)\}_{i=1}^S$.
    $\Phi_t = \frac{1}{2} \sum_{i=1}^S \left(F_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}; \xi_i^x) + F_x(\boldsymbol{x}_t, \boldsymbol{z}_{t+1}; \xi_i^x) + \dfrac{G_x(\boldsymbol{x}_t, \boldsymbol{y}_{t+1}; \zeta_i^x) - G_x(\boldsymbol{x}_t, \boldsymbol{z}_{t+1}; \zeta_i^x)}{\nu} \right)$
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_x \Phi_t / \|\Phi_t\|$
end for

G Additional Experiments

This section provides additional experiments on finding the optimal per-parameter regularization of a 5-layer MLP with ReLU activation and the hidden layer size of 500. Following the notation in Example 2.2, we let $\boldsymbol{x} \in \mathbb{R}^d$ parameterize the regularization matrix via $\boldsymbol{W}_{\boldsymbol{x}} = \mathrm{diag}(\exp(\boldsymbol{x}))$. We also let ℓ_{val} and ℓ_{tr} be the logistic loss of the network prediction on the validation set and training set, respectively. The problem to solve has the same formulation as Example 2.2, as restated below:

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} \ell_{\mathrm{val}}(\boldsymbol{y}), \quad \text{s.t.} \quad \boldsymbol{y} \in \arg\min_{\boldsymbol{y} \in \mathbb{R}^d} \ell_{\mathrm{tr}}(\boldsymbol{y}) + \boldsymbol{y}^\top \boldsymbol{W}_{\boldsymbol{x}} \boldsymbol{y}. \tag{22}$$

The difference between Example 2.2 is that now the problem is nonsmooth nonconvex due to the use of the MLP model. We present the experiment results in Figure 2.

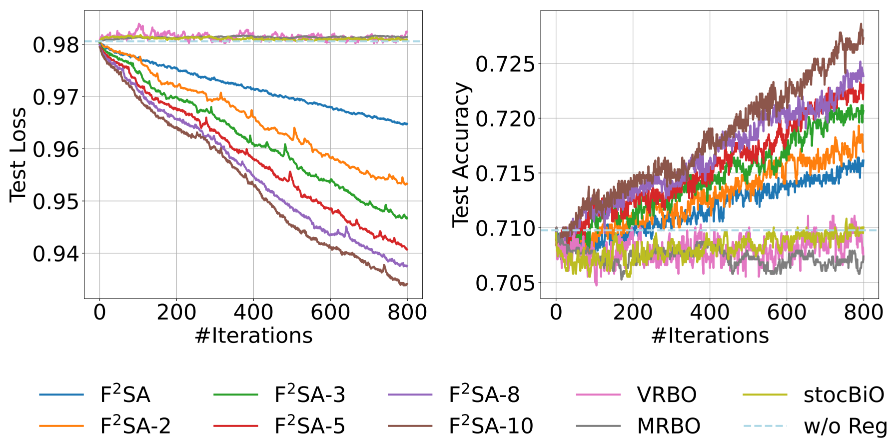

Figure 2: Performances of different algorithms on Problem (22) with an MLP model.

H Use of Large Language Models

Large language models were used to help calculate the coefficient α_0 when p is odd in Lemma 3.1, and to refine wording and correct grammatical errors in parts of the paper.