Federated Stochastic Bilevel Optimization with Fully First-Order Gradients

Yihan Zhang[1] , **Rohit Dhaipule**[2] , **Chiu C. Tan**[1] , **Haibin Ling**[2] and **Hongchang Gao**[1*]
[1]Temple University
[2]Stony Brook University
{rdhaipule, hling}@cs.stonybrook.edu, {yihan.zhang0002, chiu.tan, hongchang.gao}@temple.edu

Abstract

Federated stochastic bilevel optimization has been actively studied in recent years due to its widespread applications in machine learning. However, most existing federated stochastic bilevel optimization algorithms require the computation of second-order Hessian and Jacobian matrices, which leads to longer running times in practice. To address these challenges, we propose a novel federated stochastic variance-reduced bilevel gradient descent algorithm that relies solely on first-order oracles. Specifically, our approach does not require the computation of second-order Hessian and Jacobian matrices, significantly reducing running time. Furthermore, we introduce a novel learning rate mechanism, i.e., a constant single-timescale learning rate, to coordinate the update of different variables. We also present a new strategy to establish the convergence rate of our algorithm. Finally, the extensive experimental results confirm the efficacy of our proposed algorithm.

1 Introduction

In this paper, we focus on the following federated stochastic bilevel optimization (FedSBO) problem:

$$\min_{x\in\mathbb{R}^{d_x}} f(x,y^*(x)) \triangleq \frac{1}{N}\sum_{n=1}^{N} f^{(n)}(x,y^*(x))$$
$$s.t., y^*(x) = \arg\min_{y\in\mathbb{R}^{d_y}} g(x,y) \triangleq \frac{1}{N}\sum_{n=1}^{N} g^{(n)}(x,y)\,. \quad (1)$$

Here, $g(x,y) \triangleq \frac{1}{N}\sum_{n=1}^{N} g^{(n)}(x,y)$ denotes the lower-level loss function, where $g^{(n)}(x,y) = \mathbb{E}[g^{(n)}(x,y;\xi^{(n)})]$ denotes the lower-level loss function on the n-th device, and $f(x,y^*(x)) \triangleq \frac{1}{N}\sum_{n=1}^{N} f^{(n)}(x,y^*(x))$ represents the upper-level loss function, where $f^{(n)}(x,y^*(x)) = \mathbb{E}[f^{(n)}(x,y^*(x);\xi^{(n)})]$ represents the upper-level loss function on the n-th device. Throughout this paper, we assume that the upper-level loss function is nonconvex with respect to both variables and the lower-level loss function is strongly convex with respect to y, which is a commonly used assumption in the existing literature [Ghadimi and Wang, 2018; Ji *et al.*, 2021; Chen *et al.*, 2021; Tarzanagh *et al.*, 2022].

The stochastic bilevel optimization problem has attracted increasing attention recently because many machine learning models belong to this class of optimization problem, such as model-agnostic meta-learning [Finn *et al.*, 2017], hyperparameter optimization [Ji *et al.*, 2021], neural architecture search [Liu *et al.*, 2018], etc. To solve the stochastic bilevel optimization problem, numerous optimization algorithms [Ghadimi and Wang, 2018; Ji *et al.*, 2021; Chen *et al.*, 2021; Hong *et al.*, 2020; Khanduri *et al.*, 2021; Yang *et al.*, 2021; Guo and Yang, 2021; Dagréou *et al.*, 2022; Dagréou *et al.*, 2024] have been developed in the single machine setting in the past few years. However, enabling federated learning for the stochastic bilevel optimization problem is much more challenging than the standard single-level optimization problem. Specifically, as shown in Eq. (1), the upper-level problem *on each device* depends on the optimal solution $y^*(x)$ of the *global* lower-level problem. As a result, computing the stochastic hypergradient for the upper-level loss function on each device requires the global Jacobian and Hessian matrices, introducing significant challenges for local computation and global communication.

In recent years, numerous efforts [Gao, 2022; Tarzanagh *et al.*, 2022; Li *et al.*, 2024; Yang *et al.*, 2024; Huang *et al.*, 2023] have been made to address the unique challenges in FedSBO problems. For example, [Gao, 2022] developed a local stochastic bilevel gradient descent with momentum algorithms under the homogeneous setting, which can remove the dependence on global Jacobian and Hessian matrices due to the homogeneous data distribution. [Tarzanagh *et al.*, 2022] proposed FedNEST in the heterogeneous setting. It employs the Neumann series expansion approach in an inner loop to estimate Hessian-inverse-vector product. However, this method suffers from a large communication complexity because it requires to communicate every intermediate Hessian-inverse-vector product in that inner loop. [Li *et al.*, 2024] proposed a single-loop algorithm to address the issue of large communication complexity. In particular, it introduces an additional gradient descent procedure to replace the Neumann series expansion approach to estimate the Hessian inverse vector product.

*Corresponding author

Nevertheless, the aforementioned federated stochastic bilevel optimization algorithms still suffer from high computation costs. Specifically, those algorithms need to compute the second-order Jacobian and Hessian matrices. It is computationally expensive, especially when the problem is high-dimensional. In fact, to avoid computing the second-order Jacobian and Hessian matrices, [Kwon *et al.*, 2023; Shen and Chen, 2023; Kwon *et al.*, 2024; Chen *et al.*, 2023] developed a fully first-order method in the single-machine setting. Specifically, [Kwon *et al.*, 2023] converts the bilevel optimization problem into a single-level one, and then only the first-order gradient is required to solve it, significantly reducing computational costs. More specifically, [Kwon *et al.*, 2023] converts the lower-level optimization problem as a constraint and then leverages the penalty approach to further convert it to an unconstrained minimax optimization problem, which can be solved by the first-order stochastic gradient descent ascent algorithm. Inspired by this, we aim to develop an efficient federated stochastic bilevel optimization algorithm that uses only first-order gradients to address the aforementioned challenges. However, developing first-order methods for the FedSBO problem in Eq. (1) presents unique challenges, which are outlined below.

- First, existing first-order methods [Kwon *et al.*, 2023] for the single-machine setting employ the *iteration-dependent learning rate* to guarantee convergence. However, this strategy is not practical in federated learning. Specifically, from the iteration-dependent learning rate, the participating device is able to infer the current training stage, which could be used to attack the training procedure by attackers. Therefore, it is necessary to develop *a constant learning rate* to avoid this problem while guaranteeing convergence.
- Second, existing first-order methods [Kwon *et al.*, 2023] under the single-machine setting employ a *two-timescale learning rate* to guarantee convergence. Specifically, some variables have an order-wise larger learning rate than the others. This kind of learning rate is difficult to tune for practical federated learning practitioners. Thus, it is necessary to propose a *single-timescale learning rate* to make it easy to tune while ensuring convergence.
- Third, the first-order method under the single-machine setting requires a penalty hyperparameter to guarantee convergence. However, it is unclear how this penalty hyperparameter affects the consensus error in federated learning. Specifically, it is unclear whether the impact of the penalty hyperparameter on the consensus error will lead to a slower convergence rate. Therefore, it is necessary to establish the theoretical convergence rate, revealing how the penalty hyperparameter affects the consensus error and the convergence rate in federated learning.

To address the aforementioned unique challenges, we develop a novel federated stochastic variance-reduced bilevel gradient descent algorithm with fully first-order oracles. Specifically, on the algorithm design side, we develop a novel single-timescale constant learning rate, where we demonstrate how this new learning rate depends on the penalty hyperparameter. On the theoretical analysis side, we propose a novel strategy to establish the convergence rate of our algorithm. In particular, we develop a novel potential function for convergence analysis, where we demonstrate how to combine different components together with carefully designed coefficients. With these novel algorithmic and theoretical designs, our algorithm can achieve the $O(\frac{1}{N\epsilon^5})$ convergence rate to obtain the ϵ-accuracy solution, which indicates the linear speedup with respect to the number of devices N. Finally, extensive experimental results confirm the efficacy of our new algorithm. In summary, our paper has made the following contributions.

- We develop a novel federated stochastic variance-reduced bilevel gradient descent algorithm, which uses the fully first-order gradient and the single-timescale constant learning rate. To the best of our knowledge, this is the first time a single-timescale constant learning rate has been shown to be applied to the first-order bilevel optimization algorithms.
- We establish the convergence rate of our algorithm, where we demonstrate how to use a potential function to combine different estimation errors together with carefully designed coefficients. As far as we know, this is the first work to provide convergence guarantees for first-order federated bilevel optimization algorithms with the single-timescale constant learning rate .
- Our extensive experimental results on various tasks confirm the efficacy of our algorithm, i.e., our new algorithm is more computationally efficient than existing second-order methods.

2 Related Works

2.1 Stochastic Bilevel Optimization

Since the upper-level loss function depends on the optimal solution of the lower-level optimization problem, the hypergradient of the upper-level loss function needs to compute $\partial y^*(x)/\partial x$, which brings unique challenges for optimizing this class of optimization problems. Specifically, computing $\partial y^*(x)/\partial x$ depends on Jacobian and Hessian inverse matrices. To compute them, many efforts [Ghadimi and Wang, 2018; Ji *et al.*, 2021; Chen *et al.*, 2021; Hong *et al.*, 2020; Khanduri *et al.*, 2021; Yang *et al.*, 2021; Guo and Yang, 2021; Dagréou *et al.*, 2022; Chu *et al.*, 2024; Dagréou *et al.*, 2024] have been made in the single machine setting recently. For example, [Ghadimi and Wang, 2018] proposed using the Neumann series expansion approach to approximate the Hessian-inverse-vector product, based on which the convergence rate of the stochastic gradient descent is established. In this direction, several improved algorithms have been developed. For example, [Ji *et al.*, 2021] developed a mini-batch stochastic gradient descent method to improve the convergence rate with a large batch size. [Hong *et al.*, 2020] studied the convergence rate of the stochastic gradient descent with a two-timescale learning rate. [Chen *et al.*, 2021] established the convergence rate of the stochastic gradient descent with an alternating update strategy. To further improve the convergence rate, [Khanduri *et al.*, 2021; Yang *et al.*, 2021;

Guo and Yang, 2021] introduces the variance reduction techniques [Cutkosky and Orabona, 2019; Fang *et al.*, 2018; Nguyen *et al.*, 2017] into stochastic bilevel optimization, whose convergence rate $O(\epsilon^{-3})$ can match their counterparts for the single-level optimization problem. In addition to the Neumann series expansion approach, there exists another approach for estimating the hypergradient. Specifically, the Hessian-inverse-vector product is viewed as the optimal solution of a quadratic optimization problem, and then an additional gradient descent procedure is introduced to estimate the Hessian-inverse-vector product. Based on this approach, a couple of algorithms have been developed. For example, [Dagréou *et al.*, 2022; Chu *et al.*, 2024; Dagréou *et al.*, 2024] combined it with variance reduction techniques, whose convergence rates $O(\epsilon^{-3})$ can also match their counterparts for the single-level optimization problem.

Since the aforementioned approaches need to compute the second-order Jacobian and Hessian inverse matrices, [Kwon *et al.*, 2023; Shen and Chen, 2023] developed the fully first-order method to address this issue. In particular, they transformed the lower-level optimization problem as a constraint and then used the penalty approach to further convert it as an unconstrained minimax optimization problem. However, the penalty hyperparameter significantly affects the convergence rate. For example, [Kwon *et al.*, 2023] shows that the standard stochastic gradient descent ascent algorithm can only achieve the convergence rate of $O(\epsilon^{-7})$ and can be improved to $O(\epsilon^{-5})$ with the variance reduction technique. However, these methods do not require computation of the second-order Jacobian and Hessian inverse matrices. As a result, their practical performance is more efficient in terms of running time, especially when the problem is high-dimensional.

2.2 Federated Stochastic Bilevel Optimization

To enable distributed optimization for stochastic bilevel optimization problems, numerous algorithms [Gao, 2022; Tarzanagh *et al.*, 2022; Li *et al.*, 2024; Yang *et al.*, 2024; Huang *et al.*, 2023; Gao *et al.*, 2023; Zhang *et al.*, 2023] have recently been developed. For example, based on the Neumann series expansion approach, [Gao, 2022] developed local stochastic bilevel gradient descent with momentum algorithms under the homogeneous setting, which only need to communicate two variables and their gradient estimators. Its communication complexity $O(\epsilon^{-1})$ can match its counterpart for the single machine setting. [Tarzanagh *et al.*, 2022] proposed FedNEST for the heterogeneous setting, where every intermediate Hessian-inverse-vector product in the the Neumann series expansion approach should be communicated, leading to a large communication complexity. Later, [Huang *et al.*, 2023] developed the FedMBO algorithm, which does not need to communicate the intermediate Hessian-inverse-vector product. However, the upper-level variable in FedMBO needs to be communicated in every iteration, which also leads to a large communication complexity. Based on the other approach for estimating Hessian-inverse-vector product, [Li *et al.*, 2024] developed a federated stochastic bilevel optimization algorithm with the variance reduction technique, whose communication complexity can match that of its counterpart for the single-level problem. However, all these existing algorithms need to compute the second-order Jacobian and Hessian matrices, which is computationally expensive for high-dimensional problems. As far as we know, there do not exist first-order federated bilevel optimization algorithms. Note that a concurrent work [Yang *et al.*, 2025] appeared online after we submitted our paper to IJCAI, and it has a worse convergence rate $O(\epsilon^{-7})$.

3 Algorithm Design

3.1 Problem Definition

According to [Kwon *et al.*, 2023], Eq. (1) can be converted to a constrained optimization problem as follows:

$$\min_{x\in\mathbb{R}^{d_x},y\in\mathbb{R}^{d_y}} f(x,y),\ s.t.\ g(x,y)-g(x,y^*(x))\le 0\,. \quad (2)$$

Then, by leveraging the penalty approach, this constrained optimization problem can be converted to an unconstrained minimax optimization problem as follows:

$$\min_{x\in\mathbb{R}^{d_x},y\in\mathbb{R}^{d_y}} \max_{z\in\mathbb{R}^{d_y}} \underbrace{f(x,y)+\lambda(g(x,y)-g(x,z))}_{\mathcal{L}_\lambda(x,y,z)}\,, \quad (3)$$

where $\lambda>0$ is the penalty hyperparameter. Denoting

$$\mathcal{L}(x)=f(x,y^*(x))\,,$$
$$\mathcal{L}^*_\lambda(x)=f(x,y^*_\lambda(x))+\lambda(g(x,y^*_\lambda(x))-g(x,y^*(x)))\,,$$

where $y^*_\lambda(x)=\arg\min_{y\in\mathbb{R}^{d_y}} f(x,y)+\lambda(g(x,y)-g(x,y^*(x)))$, existing works [Kwon *et al.*, 2023] have shown that $\mathcal{L}^*_\lambda(x)$ is a good approximation to $\mathcal{L}(x)$ when given an appropriate penalty hyperparameter λ.

Based on this reformulation, we can solve Eq. (1) by optimizing the following minimax problem:

$$\min_{x,y}\max_{z}\frac{1}{N}\sum_{n=1}^{N}\underbrace{\left(f^{(n)}(x,y)+\lambda(g^{(n)}(x,y)-g^{(n)}(x,z))\right)}_{\mathcal{L}^{(n)}_\lambda(x,y,z)}\,. \quad (4)$$

Obviously, the gradients $\nabla_x\mathcal{L}^{(n)}_\lambda(x,y,z)$, $\nabla_y\mathcal{L}^{(n)}_\lambda(x,y,z)$, and $\nabla_z\mathcal{L}^{(n)}_\lambda(x,y,z)$ on the n-th device do not depend on the second-order Jacobian and Hessian matrices. Thus, Eq. (4) can be solved efficiently.

3.2 Our Algorithm

To solve Eq. (4), we develop a novel federated stochastic variance-reduced bilevel gradient descent algorithm with fully first-order oracles (FedSVRBGD-FO) in Algorithm 1. Specifically, in the t-th iteration, device n computes the stochastic variance-reduced gradient for each component of $\nabla_x\mathcal{L}^{(n)}_\lambda(x,y,z)$ as follows:

$$\begin{aligned}u^{(n)}_{1,t}&=(1-\beta_x\eta^2)(u^{(n)}_{1,t-1}-\nabla_1 f^{(n)}(x^{(n)}_{t-1},y^{(n)}_{t-1};\xi^{(n)}_t))\\&\quad+\nabla_1 f^{(n)}(x^{(n)}_t,y^{(n)}_t;\xi^{(n)}_t)\,,\\u^{(n)}_{2,t}&=(1-\beta_x\eta^2)(u^{(n)}_{2,t}-\nabla_1 g^{(n)}(x^{(n)}_{t-1},y^{(n)}_{t-1};\xi^{(n)}_t))\\&\quad+\nabla_1 g^{(n)}(x^{(n)}_t,y^{(n)}_t;\xi^{(n)}_t)\,,\end{aligned}$$

Algorithm 1 FedSVRBGD-FO

Input: $x_0, y_0, z_0, \eta > 0, \alpha_x > 0, \alpha_y > 0, \alpha_z > 0, \beta_x > 0, \beta_y > 0, \beta_z > 0$.

1: Initialization when $t = 0$: compute the stochastic gradient with a mini-batch of samples where the batch size is B:
$m_{x,0}^{(n)} = \nabla_1 f^{(n)}(x_0^{(n)}, y_0^{(n)}; \xi_0^{(n)}) + \lambda(\nabla_1 g^{(n)}(x_0^{(n)}, y_0^{(n)}; \xi_0^{(n)}) - \nabla_1 g^{(n)}(x_0^{(n)}, z_0^{(n)}; \xi_0^{(n)}))$,
$m_{y,0}^{(n)} = \nabla_2 f^{(n)}(x_0^{(n)}, y_0^{(n)}; \xi_0^{(n)}) + \lambda \nabla_2 g^{(n)}(x_0^{(n)}, y_0^{(n)}; \xi_0^{(n)})$, $m_{z,0}^{(n)} = \lambda \nabla_2 g^{(n)}(x_0^{(n)}, z_0^{(n)}; \xi_0^{(n)})$,
2: **for** $t = 0, \cdots, T-1$, each device n **do**
3: Update x: $x_{t+1}^{(n)} = x_t^{(n)} - \alpha_x \eta m_{x,t}^{(n)}$,
4: Update y : $y_{t+1}^{(n)} = y_t^{(n)} - \alpha_y \eta m_{y,t}^{(n)}$,
5: Update z: $z_{t+1}^{(n)} = z_t^{(n)} - \alpha_z \eta m_{z,t}^{(n)}$,
6: Update the variance-reduced gradient $m_{x,t+1}^{(n)}$:
$u_{1,t+1}^{(n)} = (1 - \beta_x \eta^2)(u_{1,t}^{(n)} - \nabla_1 f^{(n)}(x_t^{(n)}, y_t^{(n)}; \xi_{t+1}^{(n)})) + \nabla_1 f^{(n)}(x_{t+1}^{(n)}, y_{t+1}^{(n)}; \xi_{t+1}^{(n)})$,
$u_{2,t+1}^{(n)} = (1 - \beta_x \eta^2)(u_{2,t}^{(n)} - \nabla_1 g^{(n)}(x_t^{(n)}, y_t^{(n)}; \xi_{t+1}^{(n)})) + \nabla_1 g^{(n)}(x_{t+1}^{(n)}, y_{t+1}^{(n)}; \xi_{t+1}^{(n)})$,
$u_{3,t+1}^{(n)} = (1 - \beta_x \eta^2)(u_{3,t}^{(n)} - \nabla_1 g^{(n)}(x_t^{(n)}, z_t^{(n)}; \xi_{t+1}^{(n)})) + \nabla_1 g^{(n)}(x_{t+1}^{(n)}, z_{t+1}^{(n)}; \xi_{t+1}^{(n)})$,
$m_{x,t+1}^{(n)} = u_{1,t+1}^{(n)} + \lambda(u_{2,t+1}^{(n)} - u_{3,t+1}^{(n)})$,
7: Update the variance-reduced gradient $m_{y,t+1}^{(n)}$:
$v_{1,t+1}^{(n)} = (1 - \beta_y \eta^2)(v_{1,t}^{(n)} - \nabla_2 f^{(n)}(x_t^{(n)}, y_t^{(n)}; \xi_{t+1}^{(n)})) + \nabla_2 f^{(n)}(x_{t+1}^{(n)}, y_{t+1}^{(n)}; \xi_{t+1}^{(n)})$,
$v_{2,t+1}^{(n)} = (1 - \beta_y \eta^2)(v_{2,t}^{(n)} - \nabla_2 g^{(n)}(x_t^{(n)}, y_t^{(n)}; \xi_{t+1}^{(n)})) + \nabla_2 g^{(n)}(x_{t+1}^{(n)}, y_{t+1}^{(n)}; \xi_{t+1}^{(n)})$,
$m_{y,t+1}^{(n)} = v_{1,t+1}^{(n)} + \lambda v_{2,t+1}^{(n)}$,
8: Update the variance-reduced gradient $m_{z,t+1}^{(n)}$:
$w_{1,t+1}^{(n)} = (1 - \beta_z \eta^2)(w_{1,t}^{(n)} - \nabla_2 g^{(n)}(x_t^{(n)}, z_t^{(n)}; \xi_{t+1}^{(n)})) + \nabla_2 g^{(n)}(x_{t+1}^{(n)}, z_{t+1}^{(n)}; \xi_{t+1}^{(n)})$,
$m_{z,t+1}^{(n)} = \lambda w_{1,t+1}^{(n)}$,
9: **if** $\mathrm{mod}(t+1, p) == 0$ **then**
10: $x_{t+1}^{(n)} = \bar{x}_{t+1} = \frac{1}{N}\sum_{n'=1}^{N} x_{t+1}^{(n')}$, $y_{t+1}^{(n)} = \bar{y}_{t+1} = \frac{1}{N}\sum_{n'=1}^{N} y_{t+1}^{(n')}$, $z_{t+1}^{(n)} = \bar{z}_{t+1} = \frac{1}{N}\sum_{n'=1}^{N} z_{t+1}^{(n')}$,
$u_{1,t+1}^{(n)} = \bar{u}_{1,t+1} = \frac{1}{N}\sum_{n'=1}^{N} u_{1,t+1}^{(n')}$, $u_{2,t+1}^{(n)} = \bar{u}_{2,t+1} = \frac{1}{N}\sum_{n'=1}^{N} u_{2,t+1}^{(n')}$, $u_{3,t+1}^{(n)} = \bar{u}_{3,t+1} = \frac{1}{N}\sum_{n'=1}^{N} u_{3,t+1}^{(n')}$,
$v_{1,t+1}^{(n)} = \bar{v}_{1,t+1} = \frac{1}{N}\sum_{n'=1}^{N} v_{1,t+1}^{(n')}$, $v_{2,t+1}^{(n)} = \bar{v}_{2,t+1} = \frac{1}{N}\sum_{n'=1}^{N} v_{2,t+1}^{(n')}$, $w_{1,t+1}^{(n)} = \bar{w}_{1,t+1} = \frac{1}{N}\sum_{n'=1}^{N} w_{1,t+1}^{(n')}$,
11: **end if**
12: **end for**

$$u_{3,t}^{(n)} = (1 - \beta_x \eta^2)(u_{3,t}^{(n)} - \nabla_1 g^{(n)}(x_{t-1}^{(n)}, z_{t-1}^{(n)}; \xi_t^{(n)})) + \nabla_1 g^{(n)}(x_t^{(n)}, z_t^{(n)}; \xi_t^{(n)}) , \tag{5}$$

where $\eta > 0$ is the learning rate and $\beta_x > 0$ is a constant hyperparameter, which satisfies $\beta_x \eta^2 < 1$. Then, our algorithm composes the variance-reduced gradient estimator for $\nabla_x \mathcal{L}_\lambda^{(n)}(x, y, z)$ as follows:

$$m_{x,t}^{(n)} = u_{1,t}^{(n)} + \lambda(u_{2,t}^{(n)} - u_{3,t}^{(n)}) . \tag{6}$$

This gradient estimator is then used to update the local variable as follows:

$$x_{t+1}^{(n)} = x_t^{(n)} - \alpha_x \eta m_{x,t}^{(n)} , \tag{7}$$

where $\alpha_x > 0$ is a constant hyperparameter. Then, at every p iterations, where $p > 1$, the local variable $x_{t+1}^{(n)}$ and gradient estimators $u_{1,t+1}^{(n)}$, $u_{2,t+1}^{(n)}$, $u_{3,t+1}^{(n)}$ are uploaded to the central server and then reset to the global ones, which is shown in Step 9 in Algorithm 1. The other two variables are updated in the same way.

It is worth noting that the learning rate η and the associated hyperparameters α_x, α_y, α_z in our Algorithm 1 are constant, rather than iteration-dependent as existing methods in the single-machine setting.

4 Convergence Analysis

4.1 Assumption

To establish the convergence rate of our Algorithm 1, we introduce the following assumptions that are commonly used in existing work [Kwon *et al.*, 2023; Chen *et al.*, 2023].

Assumption 4.1. For $n \in \{1, \cdots, N\}$, the upper-level function $f^{(n)}(\cdot, \cdot)$ is L_f-smooth in expectation where $L_f > 0$ is a constant, and it is C_f-Lipschitz with respect to the second variable in expectation.

Assumption 4.2. For $n \in \{1, \cdots, N\}$, the lower-level function $g^{(n)}(\cdot, \cdot)$ is $L_{g,1}$-smooth in expectation where $L_{g,1} > 0$ is a constant and $\nabla^2 g(\cdot, \cdot)$ is $L_{g,2}$-Lipschitz in expectation where $L_{g,2} > 0$ is a constant.

Assumption 4.3. For $n \in \{1, \cdots, N\}$, the lower-level function $g^{(n)}(\cdot, \cdot)$ is μ-strongly-convex with respect to the second variable.

Based on Assumptions 4.1-4.3, [Kwon *et al.*, 2023] shows that $\mathcal{L}_\lambda(x,y,z)$ is $\frac{\lambda\mu}{2}$-strongly-convex in y when $\lambda > \frac{2L_f}{\mu}$.

Assumption 4.4. The stochastic gradients of the loss functions at the upper and lower levels have an upper bounded variance σ^2 where $\sigma > 0$ is a constant.

Assumption 4.5. The data distribution between workers is heterogeneous and the upper-level and lower-level gradients satisfy the following conditions:

$$\frac{1}{N}\sum_{n=1}^{N}\|\nabla f^{(n)}(x,y) - \frac{1}{N}\sum_{n'=1}^{N}\nabla f^{(n')}(x,y)\|^2 \le \delta^2 ,$$
$$\frac{1}{N}\sum_{n=1}^{N}\|\nabla g^{(n)}(x,y) - \frac{1}{N}\sum_{n'=1}^{N}\nabla g^{(n')}(x,y)\|^2 \le \delta^2 , \quad (8)$$

where $\delta > 0$ is a constant, $x \in \mathbb{R}^{d_x}$, and $y \in \mathbb{R}^{d_y}$.

Following [Chen *et al.*, 2023], we introduce the following definition.

Definition 4.6. Given Assumptions 4.1-4.3, we denote $\ell = \max\{C_f, L_f, L_{g,1}, L_{g,2}\}$ and $\kappa = \ell/\mu$.

With the aforementioned assumptions and definitions, the following lemma shows how well $\mathcal{L}^*_\lambda(x)$ approximates $\mathcal{L}(x)$.

Lemma 4.7. *[Chen* et al.*, 2023; Kwon* et al.*, 2023] Given Assumptions 4.1-4.3, by setting $\lambda > \frac{2L_f}{\mu}$, then $\mathcal{L}^*_\lambda(x)$ satisfies the following conditions:*

- *$\nabla\mathcal{L}^*_\lambda(x)$ is L-Lipschitz, where $L = O(\ell\kappa^3)$;*
- *$\|\nabla\mathcal{L}^*_\lambda(x) - \nabla\mathcal{L}(x)\| = O(\ell\kappa^3/\lambda)$ for any $x \in \mathbb{R}^{d_x}$;*
- *$|\mathcal{L}^*_\lambda(x) - \mathcal{L}(x)| = O(\ell\kappa^2/\lambda)$ for any $x \in \mathbb{R}^{d_x}$.*

4.2 Convergence Rate

Based on Assumptions 4.1-4.5, we established the convergence rate of our algorithm below, whose proof can be found in Appendix.

Theorem 4.8. *Given Assumptions 4.1-4.5, by setting $\lambda = O\left(\frac{\kappa^3}{\epsilon}\right)$, $\beta_x = O\left(\frac{1}{N}\right)$, $\beta_y = O\left(\frac{1}{N}\right)$, $\beta_z = O\left(\frac{1}{N}\right)$, $\alpha_x = O\left(\frac{1}{\lambda\kappa^3}\right)$, $\alpha_y = O\left(\frac{1}{\lambda\kappa}\right)$, $\alpha_z = O\left(\frac{1}{\lambda\kappa}\right)$, $\eta = O\left(\frac{N\epsilon^2}{\kappa^4}\right)$, $p = O\left(\frac{\kappa}{N\epsilon}\right)$, $B = O\left(\frac{\kappa^6}{N\epsilon^3}\right)$, and $T = O\left(\frac{\kappa^{10}}{N\epsilon^5}\right)$, Algorithm 1 can achieve the ϵ-accuracy solution as follows:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\bar{x}_t)\|^2] \le O\left(\epsilon^2\right) . \quad (9)$$

Remark 4.9. The complexity of the iteration $T = O\left(\frac{\kappa^{10}}{N\epsilon^5}\right)$ indicates that our algorithm can achieve a linear speedup with respect to the number of devices N. To the best of our knowledge, this is the first time that the first-order algorithm can achieve the linear speedup for federated bilevel optimization problems. Moreover, when $N = 1$, i.e., the single-machine setting, our convergence rate can match that of the single-machine counterpart in [Kwon *et al.*, 2023]. Furthermore, the communication complexity of Algorithm 1 is $\frac{T}{p} = O\left(\frac{\kappa^9}{\epsilon^4}\right)$.

Remark 4.10. The existing method [Kwon *et al.*, 2023] under the single-machine setting employs a time-dependent learning rate. For example, the learning rate in the t-th iteration for x is of the order of $O(t^{-3/5})$. On the contrary, our Algorithm 1 employs a constant learning rate η that is independent of the current iteration. Moreover, [Kwon *et al.*, 2023] employs a two-timescale learning rate. Specifically, the learning rate for x is $O(t^{-3/5})$, while that for z is $O(t^{-2/5})$. Then, $\lim_{t\to\infty} t^{-3/5}/t^{-2/5} = 0$. In contrast, the learning rate for three variables in Algorithm 1, i.e., $\alpha_x\eta$, $\alpha_y\eta$, and $\alpha_z\eta$, are in the same timescale [1]. Therefore, our learning rate setting is easier to tune in practice.

4.3 Proof Sketch

To establish the convergence of Algorithm 1, we develop a novel potential function as follows:

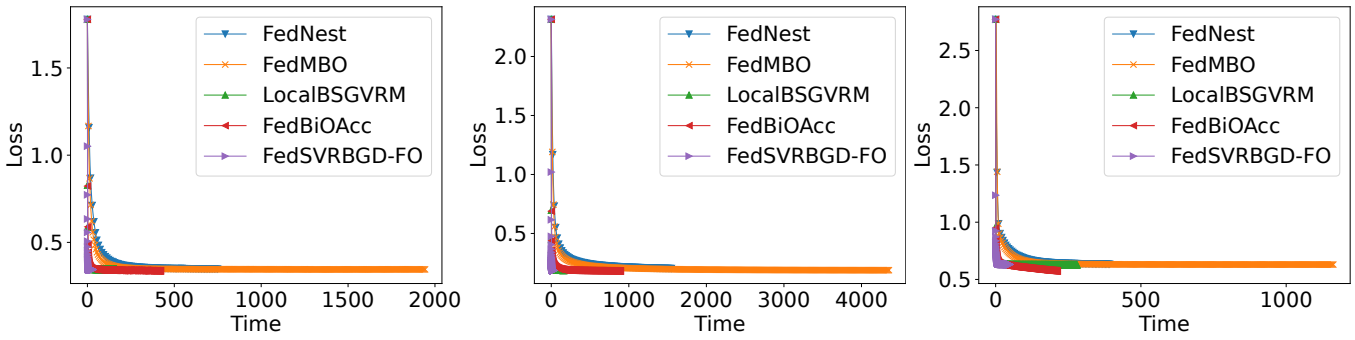
$$\begin{aligned}
P_t = {} & \mathbb{E}[\mathcal{L}^*_\lambda(\bar{x}_t)] + c_1\lambda\mathbb{E}[\|\bar{y}_t - y^*_\lambda(\bar{x}_t)\|^2] + c_2\lambda\mathbb{E}[\|\bar{z}_t - y^*(\bar{x}_t)\|^2] \\
& + c_3\mathbb{E}[\|\frac{1}{N}\sum_{n=1}^{N}\nabla_1 f^{(n)}(x_t^{(n)}, y_t^{(n)}) - \frac{1}{N}\sum_{n=1}^{N}u_{1,t}^{(n)}\|^2] \\
& + c_4\lambda^2\mathbb{E}[\|\frac{1}{N}\sum_{n=1}^{N}\nabla_1 g^{(n)}(x_t^{(n)}, y_t^{(n)}) - \frac{1}{N}\sum_{n=1}^{N}u_{2,t}^{(n)}\|^2] \\
& + c_5\lambda^2\mathbb{E}[\|\frac{1}{N}\sum_{n=1}^{N}\nabla_1 g^{(n)}(x_t^{(n)}, z_t^{(n)}) - \frac{1}{N}\sum_{n=1}^{N}u_{3,t}^{(n)}\|^2] \\
& + c_6\mathbb{E}[\|\frac{1}{N}\sum_{n=1}^{N}\nabla_2 f^{(n)}(x_t^{(n)}, y_t^{(n)}) - \frac{1}{N}\sum_{n=1}^{N}v_{1,t}^{(n)}\|^2] \\
& + c_7\lambda^2\mathbb{E}[\|\frac{1}{N}\sum_{n=1}^{N}\nabla_2 g^{(n)}(x_t^{(n)}, y_t^{(n)}) - \frac{1}{N}\sum_{n=1}^{N}v_{2,t}^{(n)}\|^2] \\
& + c_8\lambda^2\mathbb{E}[\|\frac{1}{N}\sum_{n=1}^{N}\nabla_2 g^{(n)}(x_t^{(n)}, z_t^{(n)}) - \frac{1}{N}\sum_{n=1}^{N}w_{1,t}^{(n)}\|^2] ,
\end{aligned} \quad (10)$$

where $\{c_i\}_{i=1}^8$ are positive coefficients.

A key design in our potential function is that c_1 and c_2 are associated with λ while c_4, c_5, c_7, and c_8 are associated with λ^2. As such, **all coefficients $\{c_i\}_{i=1}^8$ are independent of the penalty parameter λ. Otherwise, $\{c_i\}_{i=1}^8$ will depend on λ, which will further affect the variance term and then result in a worse convergence rate**. For example, as shown in Eq. (68), the coefficient $\{c_i\}_{i=3}^8$ affects the terms regarding the variance in the last line. If they depended on the penalty parameter, the learning rate η has to be much smaller to control the variance term. For example, when $c_4 = O(\lambda^2)$, η should be as small as $O(\epsilon^6)$ because of $\lambda = O(\frac{1}{\epsilon})$, leading to a much slower convergence rate.

Based on this potential function, we first establish the upper bound for each term of this potential function in the Appendix A.1 and then identify the coefficient $\{c_i\}_{i=1}^8$ in Appendix A.2 to eliminate all their associated terms in the upper bound of $P_{t+1} - P_t$, resulting in Eq. (131). Then, we can establish the convergence rate of our Algorithm 1.

[1] In this paper, the same timescale refers to having the same order with respect to ϵ or the number of iterations.



(a) a9a (b) w8a (c) covtype

Figure 1: The upper-level loss function value versus the running time (seconds). Communication period $p = 4$.

Moreover, a key step for establishing the final convergence rate is to bound the consensus error in the following lemma, whose proof can be found in Appendix A.1.

Lemma 4.11. *Given Assumptions 4.1-4.5, and* $\eta \leq \frac{1}{500p\sqrt{L_f^2+L_{g,1}^2\lambda^2}}$, $\beta_x \leq \frac{L_f^2+L_{g,1}^2\lambda^2}{N}$, $\beta_y \leq \frac{L_f^2+L_{g,1}^2\lambda^2}{N}$, $\beta_z \leq \frac{L_f^2+L_{g,1}^2\lambda^2}{N}$, $\alpha_x \leq 10$, $\alpha_y \leq 10$, $\alpha_z \leq 10$, *we have*

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{N}\sum_{n=1}^{N}\Big(\mathbb{E}[\|u_{1,t}^{(n)}-\bar{u}_{1,t}\|^2]+\lambda^2\mathbb{E}[\|u_{2,t}^{(n)}-\bar{u}_{2,t}\|^2]\\
&\quad+\lambda^2\mathbb{E}[\|u_{3,t}^{(n)}-\bar{u}_{3,t}\|^2]+\mathbb{E}[\|v_{1,t}^{(n)}-\bar{v}_{1,t}\|^2]\\
&\quad+\lambda^2\mathbb{E}[\|v_{2,t}^{(n)}-\bar{v}_{2,t}\|^2]+\lambda^2\mathbb{E}[\|w_{1,t}^{(n)}-\bar{w}_{1,t}\|^2]\Big)\\
&\leq 576p^2\alpha_x^2\eta^2\Big(L_f^2+L_{g,1}^2\lambda^2\Big)\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\bar{m}_{x,t}\|^2]\\
&\quad+576p^2\alpha_y^2\eta^2\Big(L_f^2+L_{g,1}^2\lambda^2\Big)\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\bar{m}_{y,t}\|^2]\\
&\quad+576p^2\alpha_z^2\eta^2\Big(L_f^2+L_{g,1}^2\lambda^2\Big)\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\bar{m}_{z,t}\|^2]\\
&\quad+576p^2\beta_x^2\eta^4(1+\lambda^2)\delta^2+576p^2\beta_y^2\eta^4(1+\lambda^2)\delta^2\\
&\quad+576p^2\beta_x^2\eta^4(1+\lambda^2)\sigma^2+576p^2\beta_y^2\eta^4(1+\lambda^2)\sigma^2\\
&\quad+576p^2\beta_z^2\eta^4\lambda^2\delta^2+576p^2\beta_z^2\eta^4\lambda^2\sigma^2\,. \qquad (11)
\end{aligned}$$

In this lemma, we reveal how the penalty hyperparameter λ and the communication period p affect the consensus error. With such a lemma, we can then establish the convergence rate of our Algorithm 1.

5 Experiments

In this section, we evaluate the performance of our Algorithm 1 on the commonly used benchmark task: hyperparameter optimization and hyper-representation learning.

5.1 Hyperparameter Optimization

In the experiment, we apply Algorithm 1 to the following hyperparameter optimization problem:

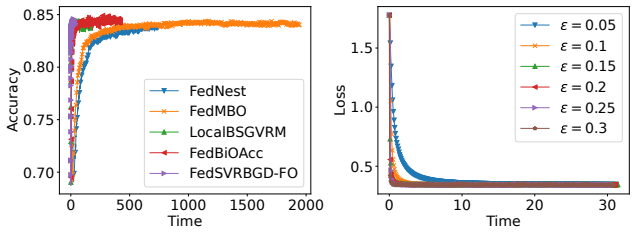
$$\begin{aligned}
&\min_{x\in\mathbb{R}^d}\frac{1}{N}\sum_{n=1}^{N}\frac{1}{m^{(n)}}\sum_{i=1}^{m^{(n)}}\ell(y^*(x)^T a_{v,i}^{(n)}, b_{v,i}^{(n)})\\
&s.t.\ y^*(x)=\arg\min_{y\in\mathbb{R}^{d\times c}}\frac{1}{N}\sum_{n=1}^{N}\frac{1}{m^{(n)}}\sum_{i=1}^{m^{(n)}}\ell(y^T a_{t,i}^{(n)}, b_{t,i}^{(n)}) \qquad (12)\\
&\qquad\qquad+\frac{1}{cd}\sum_{p=1}^{c}\sum_{q=1}^{d}\exp(x_q)y_{pq}^2\,,
\end{aligned}$$

where the lower-level problem is to learn the parameter $y \in \mathbb{R}^{d\times c}$ of the logistic regression model with the training set $\{(a_{t,i}^{(n)}, b_{t,i}^{(n)})\}_{i=1}^{m^{(n)}}$, the upper-level problem learns the the regularization coefficient $x \in \mathbb{R}^d$ with the validation set $\{(a_{v,i}^{(n)}, b_{v,i}^{(n)})\}_{i=1}^{m^{(n)}}$.

In this experiment, we used three benchmark datasets: a9a, w8a, covtype, which are obtained from LIBSVM datasets [2]. Here, 10% of the samples are randomly selected as the test set. For the remaining samples, 70% of them are randomly selected as the training set and the others are used as the validation set. The training and validation sets are then randomly distributed to eight workers. The batch size for each worker is set to 10 in this experiment. To verify the performance of our Algorithm 1, we compare it with four state-of-the-art second-order methods: FedNEST [Tarzanagh *et al.*, 2022], FedMBO [Huang *et al.*, 2023], LocalBSGVRM [Gao, 2022], and FedBiOAcc [Li *et al.*, 2024]. In our experiment, we set the solution accuracy ϵ to 0.1. Then, according to the theoretical results in [Tarzanagh *et al.*, 2022; Huang *et al.*, 2023; Gao, 2022; Li *et al.*, 2024], the learning rate of FedNEST and FedMBO is set to ϵ^2, while that of LocalBSGVRM and FedBiOAcc is set to ϵ. Regarding our method, according to Theorem 4.8, we set the learning rate η to ϵ^2, the coefficient $\alpha_x = \alpha_y = \alpha_z = \epsilon$, and the penalty $\lambda = 5/\epsilon$. Moreover, for all methods using the momentum-based variance reduction technique, we set the the coefficient of the momentum to 0.1. Then, we run all experiments on a workstation with 4 NVIDIA A5000 GPU cards, each of which accommodates two threads to simulate eight workers.

[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

In Figure 1, we plot the upper-level loss function value versus the running time (seconds). Specifically, to make a fair comparison, we ran all methods to update x for 16,000 iterations on all datasets. The communication period is set to 4 in this experiment. From Figure 1, we can find that our algorithm requires much less time to converge than all baseline methods. This confirms the advantage of using the fully first-order gradient in practical applications.



(a) Test Performance (b) Different ϵ

Figure 2: (a) shows the test accuracy versus the running time (seconds). (b) shows the upper-level loss function value when using different ϵ. Both use a9a dataset.

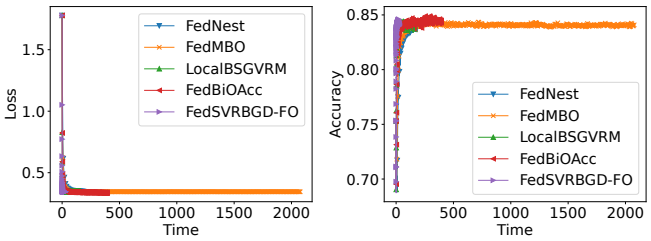
In addition, in Figure 2(a), we plot the accuracy in the test set versus the running time for the a9a dataset. In Figure 2(a), we can still find that our algorithm uses much less time to achieve almost the same accuracy as the baseline methods, further confirming the efficacy of using fully first-order gradients for federated bilevel optimization.

Because the learning rate and penalty parameter are set according to the accuracy of the solution ϵ as suggested by Theorem 4.8, we added an additional experiment to show the influence of ϵ on the convergence rate, which is shown in Figure 2(b). It can be observed that a smaller ϵ leads to a slower convergence rate. The reason is that a smaller ϵ results in a smaller learning rate as shown in Theorem 4.8, slowing the convergence.

Furthermore, to verify the impact of the communication period on convergence performance, we performed an additional experiment with $p = 16$. In Figure 3, we plot the upper-level loss function value versus the running time in Figure 3(a) and plot the test accuracy in Figure 3(b). Note that FedNest becomes more efficient than FedBiOAcc because it computes the second-order Hessian and Jacobian matrices in each communication round, rather than each iteration. Thus, it computes second-order information less frequently when the communication period becomes large. However, we can still find that our algorithm needs less time to converge than all baseline methods when the communication period is 16.

5.2 Hyper-representation Learning

To further verify the performance of our algorithm, we apply our algorithm to the hyper-representation learning task as [Tarzanagh *et al.*, 2022]. Specifically, given a deep neural network, the upper-level problem learns the weight of hidden layers, while the lower-level problem learns the weight of the classifier. As such, the upper-level problem is nonconvex and the lower-level one is strongly convex when using



(a) Loss (b) Accuracy

Figure 3: The upper-level loss function value and the test accuracy versus the running time (seconds) for a9a dataset. Communication period $p = 16$.

the cross-entropy loss function with ℓ_2-norm regularization for the classifier's weight. In our experiment, we used a fully connected two-layer neural network. Its dimensionality of the input, hidden, and output layer is 54, 30, and 7, respectively. The dataset is covtype, which has seven classes. The batch size is 100. The communication period $p = 4$. All the other settings are the same as the first experiment.

In Figure 4, we plot the loss function value versus the running time (seconds) for the hyper-representation task. It can be seen that our algorithm converges much faster than all baseline methods in terms of running time, which further confirms the advantages of using the first-order gradient over the second-order ones.

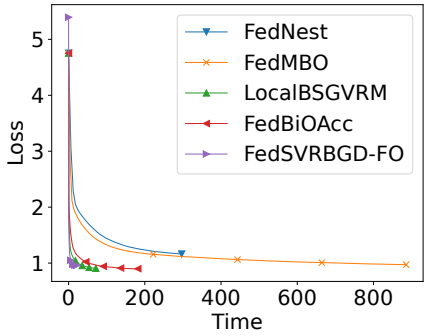


Figure 4: The upper-level loss function value versus the running time (seconds) for the hyper-representation task.

6 Conclusions

In this paper, we develop a novel federated stochastic bilevel optimization algorithm based on the fully first-order gradient. In particular, each work only needs to compute the first-order stochastic variance-reduced gradient to update the upper- and lower-level variables. This can significantly save running time because our algorithm does not need to compute the second-order Hessian and Jacobian matrices. Moreover, we developed a novel single-timescale constant learning rate to coordinate the update of different variables and presented a novel strategy to establish the convergence rate of our algorithm. The extensive experimental results confirm the efficacy of our algorithm.

Acknowledgments

Y. Zhang and H. Gao were partially supported by U.S. NSF CAREER 2339545, NSF IIS 2416607, NSF CNS 2107014.

References

[Chen *et al.*, 2021] Tianyi Chen, Yuejiao Sun, and Wotao Yin. Tighter analysis of alternating stochastic gradient method for stochastic nested problems. *arXiv preprint arXiv:2106.13781*, 2021.

[Chen *et al.*, 2023] Lesi Chen, Yaohua Ma, and Jingzhao Zhang. Near-optimal fully first-order algorithms for finding stationary points in bilevel optimization. *arXiv preprint arXiv:2306.14853*, 2023.

[Chu *et al.*, 2024] Tianshu Chu, Dachuan Xu, Wei Yao, and Jin Zhang. Spaba: A single-loop and probabilistic stochastic bilevel algorithm achieving optimal sample complexity. *arXiv preprint arXiv:2405.18777*, 2024.

[Cutkosky and Orabona, 2019] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32, 2019.

[Dagréou *et al.*, 2022] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. *arXiv preprint arXiv:2201.13409*, 2022.

[Dagréou *et al.*, 2024] Mathieu Dagréou, Thomas Moreau, Samuel Vaiter, and Pierre Ablin. A lower bound and a near-optimal algorithm for bilevel empirical risk minimization. In *International Conference on Artificial Intelligence and Statistics*, pages 82–90. PMLR, 2024.

[Fang *et al.*, 2018] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in Neural Information Processing Systems*, 31, 2018.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[Gao *et al.*, 2023] Hongchang Gao, Bin Gu, and My T Thai. On the convergence of distributed stochastic bilevel optimization algorithms over a network. In *International Conference on Artificial Intelligence and Statistics*, pages 9238–9281. PMLR, 2023.

[Gao, 2022] Hongchang Gao. On the convergence of momentum-based algorithms for federated stochastic bilevel optimization problems. *arXiv preprint arXiv:2204.13299*, 2022.

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

[Guo and Yang, 2021] Zhishuai Guo and Tianbao Yang. Randomized stochastic variance-reduced methods for stochastic bilevel optimization. *arXiv e-prints*, pages arXiv–2105, 2021.

[Hong *et al.*, 2020] Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

[Huang *et al.*, 2023] Minhui Huang, Dewei Zhang, and Kaiyi Ji. Achieving linear speedup in non-iid federated bilevel learning. *arXiv preprint arXiv:2302.05412*, 2023.

[Ji *et al.*, 2021] Kaiyi Ji, Junjie Yang, and Yingbin Liang. Bilevel optimization: Convergence analysis and enhanced design. In *International Conference on Machine Learning*, pages 4882–4892. PMLR, 2021.

[Khanduri *et al.*, 2021] 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. *Advances in Neural Information Processing Systems*, 34, 2021.

[Kwon *et al.*, 2023] Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, and Robert D Nowak. A fully first-order method for stochastic bilevel optimization. In *International Conference on Machine Learning*, pages 18083–18113. PMLR, 2023.

[Kwon *et al.*, 2024] Jeongyeol Kwon, Dohyun Kwon, and Hanbaek Lyu. On the complexity of first-order methods in stochastic bilevel optimization. *arXiv preprint arXiv:2402.07101*, 2024.

[Li *et al.*, 2024] Junyi Li, Feihu Huang, and Heng Huang. Communication-efficient federated bilevel optimization with global and local lower level problems. *Advances in Neural Information Processing Systems*, 36, 2024.

[Liu *et al.*, 2018] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[Nguyen *et al.*, 2017] Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621. PMLR, 2017.

[Shen and Chen, 2023] Han Shen and Tianyi Chen. On penalty-based bilevel gradient descent method. In *International Conference on Machine Learning*, pages 30992–31015. PMLR, 2023.

[Tarzanagh *et al.*, 2022] Davoud Ataee Tarzanagh, Mingchen Li, Christos Thrampoulidis, and Samet Oymak. Fednest: Federated bilevel, minimax, and compositional optimization. In *International Conference on Machine Learning*, pages 21146–21179. PMLR, 2022.

[Yang *et al.*, 2021] Junjie Yang, Kaiyi Ji, and Yingbin Liang. Provably faster algorithms for bilevel optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

[Yang *et al.*, 2024] Yifan Yang, Peiyao Xiao, and Kaiyi Ji. Simfbo: Towards simple, flexible and communication-

efficient federated bilevel learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[Yang *et al.*, 2025] Yifan Yang, Peiyao Xiao, Shiqian Ma, and Kaiyi Ji. First-order federated bilevel learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 22029–22037, 2025.

[Zhang *et al.*, 2023] Yihan Zhang, My T Thai, Jie Wu, and Hongchang Gao. On the communication complexity of decentralized bilevel optimization. *arXiv preprint arXiv:2311.11342*, 2023.