Provably Convergent Federated Trilevel Learning

Yang Jiao[1], **Kai Yang**[1,2,3*], **Tiancheng Wu**[1], **Chengtao Jian**[1], **Jianwei Huang**[4,5]

[1]Department of Computer Science and Technology, Tongji University
[2] Key Laboratory of Embedded System and Service Computing Ministry of Education at Tongji University
[3]Shanghai Research Institute for Intelligent Autonomous Systems
[4]School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen
[5]Shenzhen Institute of Artificial Intelligence and Robotics for Society
yangjiao@tongji.edu.cn, kaiyang@tongji.edu.cn, tony318@tongji.edu.cn, jct@tongji.edu.cn, jianweihuang@cuhk.edu.cn

Abstract

Trilevel learning, also called trilevel optimization (TLO), has been recognized as a powerful modelling tool for hierarchical decision process and widely applied in many machine learning applications, such as robust neural architecture search, hyperparameter optimization, and domain adaptation. Tackling TLO problems has presented a great challenge due to their nested decision-making structure. In addition, existing works on TLO face the following key challenges: 1) they all focus on the non-distributed setting, which may lead to privacy breach; 2) they do not offer any non-asymptotic convergence analysis which characterizes how fast an algorithm converges. To address the aforementioned challenges, this paper proposes an asynchronous federated trilevel optimization method to solve TLO problems. The proposed method utilizes μ-cuts to construct a hyper-polyhedral approximation for the TLO problem and solve it in an asynchronous manner. We demonstrate that the proposed μ-cuts are applicable to not only convex functions but also a wide range of non-convex functions that meet the μ-weakly convex assumption. Furthermore, we theoretically analyze the non-asymptotic convergence rate for the proposed method by showing its iteration complexity to obtain ϵ-stationary point is upper bounded by $\mathcal{O}(\frac{1}{\epsilon^2})$. Extensive experiments on real-world datasets have been conducted to elucidate the superiority of the proposed method, e.g., it has a faster convergence rate with a maximum acceleration of approximately 80%.

Introduction

Recently, trilevel learning, also called trilevel optimization (TLO), has found applications in many machine learning tasks, e.g., robust neural architecture search (Guo et al. 2020), robust hyperparameter optimization (Sato, Tanaka, and Takeda 2021) and domain adaptation (Raghu et al. 2021). Trilevel optimization problems refer to the optimization problems that involve three-level optimization problems and thus have a trilevel hierarchy (Avraamidou 2018; Sato, Tanaka, and Takeda 2021). A general form of trilevel optimization problem is given by,

$$\begin{aligned} &\min f_1(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \text{ s.t.} \\ &\quad \boldsymbol{x}_2 = \arg\min_{\boldsymbol{x}_2{}'} f_2(\boldsymbol{x}_1, \boldsymbol{x}_2{}', \boldsymbol{x}_3) \text{ s.t.} \\ &\qquad \boldsymbol{x}_3 = \arg\min_{\boldsymbol{x}_3{}'} f_3(\boldsymbol{x}_1, \boldsymbol{x}_2{}', \boldsymbol{x}_3{}') \\ &\text{var.} \qquad \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \end{aligned} \tag{1}$$

where f_1, f_2, f_3 respectively denote the first, second, and third level objectives. Here $\boldsymbol{x}_1 \in \mathbb{R}^{d_1}, \boldsymbol{x}_2 \in \mathbb{R}^{d_2}, \boldsymbol{x}_3 \in \mathbb{R}^{d_3}$ are variables. Despite its wide applications, the development of solution methods was predominately limited to bilevel optimization (BLO) (Ji, Yang, and Liang 2021; Franceschi et al. 2018) primarily due to the escalated difficulty in solving the TLO problem (Sato, Tanaka, and Takeda 2021). The literature, specifically (Blair 1992; Avraamidou 2018), highlights that the complexity associated with solving problems characterized by hierarchical structures comprising more than two levels is substantially greater compared to that of bilevel optimization problems.

Theoretical work on solving TLO problems only emerge during the recent several years. A hypergradient (gradient)-based method is proposed in (Sato, Tanaka, and Takeda 2021), which uses K gradient descent steps to replace the lower-level problem to solve the TLO problems. This algorithm in (Sato, Tanaka, and Takeda 2021) is one of the first results that establish theoretical guarantees for solving the TLO problem. A general automatic differentiation technique is proposed in (Choe et al. 2022), which is based on the interpretation of TLO as a special type of dataflow graph. Nevertheless, there are still some issues that have not been addressed in the prior work, including 1) in TLO applications, data may be acquired and disseminated across multiple nodes, the prior works only solve the TLO problems in a non-distributed manner, which needs to collect a massive amount of data to a single server and may lead to the data privacy risks (Subramanya and Riggio 2021; Jiao et al. 2022b; Han, Wang, and Leung 2020). Moreover, the synchronous federated algorithms often suffer from straggler problems and will immediately stop working if some workers fail to communicate (Jiao et al. 2022b). Therefore, developing asynchronous federated algorithms for TLO is significantly important. 2) The existing TLO works only provide the asymptotic convergence guarantee for their algorithms.

*Corresponding author (e-mail: kaiyang@tongji.edu.cn).

In order to understand the convergence speed of the proposed algorithm, non-asymptotic convergence analysis that can characterize how fast an algorithm converges in practice is required.

To this end, we propose an **A**synchronous **F**ederated **T**rilevel **O**ptimization method (AFTO) in this paper. The proposed AFTO can effectively solve the TLO problems in an asynchronous federated manner. Specifically, it treats the lower-level optimization problem as a constraint to the upper-level and utilizes μ-cuts to construct the hyper-polyhedral approximation, then an effective asynchronous algorithm is developed. In the context of trilevel learning problems, the objective functions at each level are usually non-convex, thus the cutting plane methods tailored for convex functions (Jiao et al. 2022b; Franc, Sonnenburg, and Werner 2011) are found to be inapplicable. To our best knowledge, the proposed methodology referred to as μ-cut represents the first approach that is capable of constructing cutting planes for trilevel learning problems characterized by non-convex objectives. Furthermore, we demonstrate that the proposed method is guaranteed to converge and theoretically analyze the non-asymptotic convergence rate in terms of iteration complexity.

The contributions of this work are summarized as follows.

1. An asynchronous federated trilevel optimization method is proposed in this work for trilevel learning. To our best knowledge, it is the first work designing algorithms to solve the trilevel learning problem in an asynchronous distributed manner.

2. A novel hyper-polyhedral approximation method via μ-cut is proposed in this work. The proposed μ-cut can be applied to trilevel learning with non-convex objectives. We further demonstrate that the iteration complexity of the proposed method to achieve the ϵ-stationary point is upper bounded by $\mathcal{O}(\frac{1}{\epsilon^2})$.

3. Extensive experiments on real-world datasets justify the superiority of the proposed method and underscore the significant benefits of employing the hyper-polyhedral approximation for trilevel learning.

Related Work

Trilevel Optimization

Trilevel optimization has many applications ranging from economics to machine learning. A robust neural architecture search approach is proposed in (Guo et al. 2020), which integrates the adversarial training into one-shot neural architecture search and can be regarded as solving a trilevel optimization problem. TimeAutoAD (Jiao et al. 2022a) is proposed to automatically configure the anomaly detection pipeline and optimize the hyperparameters for multivariate time series anomaly detection. The optimization problem that TimeAutoAD aims to solve can be viewed as a trilevel optimization problem. And a method is proposed in (Raghu et al. 2021) to solve the trilevel optimization problem which involves hyperparameter optimization and two-level pretraining and finetuning. LFM (Garg et al. 2022) is proposed to solve a trilevel optimization problem which consists of data reweight, architecture search, and model training. A general automatic differentiation technique Betty is proposed in (Choe et al. 2022), which can be utilized to solve the trilevel optimization problem. However, the aforementioned algorithms do not provide any convergence guarantee. A hypergradient-based algorithm with asymptotic convergence guarantee is proposed in (Sato, Tanaka, and Takeda 2021), which can be employed in trilevel optimization problems. Nevertheless, the existing works focus on solving the TLO problems in a non-distributed manner and do not provide any non-asymptotic convergence analysis. Instead, an efficient asynchronous algorithm with non-asymptotic convergence guarantee is proposed in this work for solving TLO problems. To our best knowledge, this is the first work that solves TLO problems in an asynchronous federated manner.

Polyhedral Approximation

Polyhedral approximation is a widely-used approximation method (Bertsekas 2015). The idea behind polyhedral approximation is to approximate either the feasible region or the epigraph of the objective function of an optimization problem by a set of cutting planes, and the approximation will be gradually refined by adding additional cutting planes. Since the approximate problem is polyhedral, it is usually much easier to solve than the original problem. Following (Bertsekas 2015), the polyhedral approximation can be broadly divided into two main approaches: outer linearization and inner linearization. The outer linearization (Tawarmalani and Sahinidis 2005; Yang et al. 2008; Bürger, Notarstefano, and Allgöwer 2013) (also called cutting plane method) utilizes a set of cutting planes to approximate the feasible region or the epigraph of the objective function from without. In contrast, inner linearization (Bertsekas and Yu 2011; Trombettoni et al. 2011) utilizes the convex hulls of finite numbers of halflines or points to approximate the feasible region or the epigraph of the objective function from within. Polyhedral approximation has been widely used in convex optimization. A polyhedral approximation method is proposed in (Bertsekas 2015) for convex optimization, which utilizes cutting planes to approximate the original convex optimization problem. In (Bürger, Notarstefano, and Allgöwer 2013), a fully distributed algorithm is proposed, which is based on an outer polyhedral approximation of the constraint sets, for the convex and robust distributed optimization problems in peer-to-peer networks. In this work, a novel hyper-polyhedral approximation method via μ-cut is proposed for TLO. The proposed μ-cut can be utilized for μ-weakly convex optimization and thus has broader applicability compared with the cutting plane methods for convex optimization.

Asynchronous Federated Trilevel Learning

Traditional trilevel optimization methods require collecting a massive amount of data to a single server for model training, which may lead to data privacy risks. Solving trilevel optimization problems in a distributed manner is challenging since the trilevel optimization problem is highly-nested which hinders the development of the distributed algorithms. The distributed trilevel optimization problem can be ex-

pressed as,

$$\begin{aligned}&\min \textstyle\sum_{j=1}^{N} f_{1,j}(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3) \text{ s.t.}\\ &\quad \boldsymbol{x}_2=\mathop{\arg\min}_{\boldsymbol{x}_2'} \textstyle\sum_{j=1}^{N} f_{2,j}(\boldsymbol{x}_1,\boldsymbol{x}_2',\boldsymbol{x}_3) \text{ s.t.}\\ &\quad\quad \boldsymbol{x}_3=\mathop{\arg\min}_{\boldsymbol{x}_3'} \textstyle\sum_{j=1}^{N} f_{3,j}(\boldsymbol{x}_1,\boldsymbol{x}_2',\boldsymbol{x}_3')\\ &\text{var.}\qquad\quad \boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\end{aligned} \tag{2}$$

where N denotes the number of workers in distributed systems, $f_{1,j}, f_{2,j}, f_{3,j}$ denote the local first, second, and third level objectives in worker j, respectively. The problem in Eq. (2) can be reformulated as a consensus problem (Zhang and Kwok 2014; Jiao, Yang, and Song 2022),

$$\begin{aligned}&\min \textstyle\sum_j f_{1,j}(\boldsymbol{x}_{1,j},\boldsymbol{x}_{2,j},\boldsymbol{x}_{3,j}) \text{ s.t.}\\ &\boldsymbol{x}_{1,j}=\boldsymbol{z}_1, j=1,\cdots,N\\ &\{\boldsymbol{x}_{2,j}\},\boldsymbol{z}_2=\mathop{\arg\min}_{\{\boldsymbol{x}_{2,j'}\},\boldsymbol{z}_2'} \textstyle\sum_j f_{2,j}(\boldsymbol{z}_1,\boldsymbol{x}_{2,j}',\boldsymbol{x}_{3,j}) \text{ s.t.}\\ &\quad \boldsymbol{x}_{2,j}'=\boldsymbol{z}_2', j=1,\cdots,N\\ &\quad \{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_3=\mathop{\arg\min}_{\{\boldsymbol{x}_{3,j'}\},\boldsymbol{z}_3'} \textstyle\sum_j f_{3,j}(\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{x}_{3,j}') \text{ s.t.}\\ &\quad\quad \boldsymbol{x}_{3,j}'=\boldsymbol{z}_3', j=1,\cdots,N\\ &\text{var.}\qquad \{\boldsymbol{x}_{1,j}\},\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3,\end{aligned} \tag{3}$$

where $\boldsymbol{x}_{1,j}\in\mathbb{R}^{d_1}, \boldsymbol{x}_{2,j}\in\mathbb{R}^{d_2}, \boldsymbol{x}_{3,j}\in\mathbb{R}^{d_3}$ denote the local variables in worker j, and $\boldsymbol{z}_1\in\mathbb{R}^{d_1}, \boldsymbol{z}_2\in\mathbb{R}^{d_2}, \boldsymbol{z}_3\in\mathbb{R}^{d_3}$ denote the consensus variables in the master. This reformulation in Eq. (3) can facilitate the development of distributed algorithms for trilevel optimization problems based on the parameter-server architecture (Assran et al. 2020). The remaining procedure of the proposed method can be divided into three steps. First, how to construct the hyper-polyhedral approximation for distributed TLO problems is proposed. Then, an effective asynchronous federated algorithm is developed. Finally, how to update the μ-cuts to refine the hyper-polyhedral approximation is proposed.

Hyper-Polyhedral Approximation

Different from the traditional polyhedral approximation method (Bertsekas 2015; Franc, Sonnenburg, and Werner 2011; Bürger, Notarstefano, and Allgöwer 2013), a novel hyper-polyhedral approximation method is proposed for distributed TLO problems in this work. By utilizing the proposed hyper-polyhedral approximation, the distributed algorithms can be easier to develop for TLO problems. Specifically, the proposed hyper-polytope consists of the I$^{\text{st}}$ layer and II$^{\text{nd}}$ layer polytopes, which are introduced as follows.

I$^{\text{st}}$ layer Polyhedral Approximation: First, defining $h_{\text{I}}(\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3)=||\begin{bmatrix}\{\boldsymbol{x}_{3,j}\}\\ \boldsymbol{z}_3\end{bmatrix}-\phi_{\text{I}}(\boldsymbol{z}_1,\boldsymbol{z}_2')||^2$ and $\phi_{\text{I}}(\boldsymbol{z}_1,\boldsymbol{z}_2')=\arg\min_{\{\boldsymbol{x}_{3,j'}\},\boldsymbol{z}_3'}\{\sum_{j=1}^{N} f_{3,j}(\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{x}_{3,j}'): \boldsymbol{x}_{3,j}'=\boldsymbol{z}_3', \forall j\}$. In trilevel optimization, the third level optimization problem can be viewed as the constraint to the second level optimization problem (Chen et al. 2022a), i.e., $h_{\text{I}}(\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3)=0$. A consensus problem needs to be solved in a distributed manner if the exact $\phi_{\text{I}}(\boldsymbol{z}_1,\boldsymbol{z}_2')$ is required. In many works in bilevel (Ji, Yang, and Liang 2021; Liu et al. 2021; Jiao et al. 2022b) and trilevel (Sato, Tanaka, and Takeda 2021) optimization, the exact $\phi_{\text{I}}(\boldsymbol{z}_1,\boldsymbol{z}_2')$ can be replaced by an estimate of $\phi_{\text{I}}(\boldsymbol{z}_1,\boldsymbol{z}_2')$, and we utilize the results after K communication rounds between the master and workers as the estimate of $\phi_{\text{I}}(\boldsymbol{z}_1,\boldsymbol{z}_2')$ according to (Jiao et al. 2022b). Specifically, for the third level optimization problem, the augmented Lagrangian function can be written as,

$$\begin{aligned}L_{p,3}=&\textstyle\sum_{j=1}^{N}(f_{3,j}(\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{x}_{3,j}')+\boldsymbol{\varphi}_{3,j}^{\top}(\boldsymbol{x}_{3,j}'-\boldsymbol{z}_3')\\ &+\tfrac{\kappa_3}{2}||\boldsymbol{x}_{3,j}'-\boldsymbol{z}_3'||^2),\end{aligned} \tag{4}$$

where $L_{p,3}=L_{p,3}(\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3',\{\boldsymbol{x}_{3,j}'\},\{\boldsymbol{\varphi}_{3,j}\})$, $\boldsymbol{\varphi}_{3,j}\in\mathbb{R}^{d_3}$ is the dual variable, and constant $\kappa_3>0$ is a penalty parameter. In $(k+1)^{\text{th}}$ communication round, we have that,
1) Workers update the local variables,

$$\boldsymbol{x}_{3,j}^{k+1\prime}=\boldsymbol{x}_{3,j}^{k\ \prime}-\eta_{\boldsymbol{x}}\nabla_{\boldsymbol{x}_{3,j}}L_{p,3}(\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3^{k\prime},\{\boldsymbol{x}_{3,j}^{k\ \prime}\},\{\boldsymbol{\varphi}_{3,j}^{k}\}), \tag{5}$$

where $\eta_{\boldsymbol{x}}$ represents the step-size. Then, workers transmit the local variables $\boldsymbol{x}_{3,j}^{k+1\prime}$ to the master.
2) Master updates the variables as follows,

$$\boldsymbol{z}_3^{k+1\prime}=\boldsymbol{z}_3^{k\prime}-\eta_{\boldsymbol{z}}\nabla_{\boldsymbol{z}_3}L_{p,3}(\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3^{k\prime},\{\boldsymbol{x}_{3,j}^{k\ \prime}\},\{\boldsymbol{\varphi}_{3,j}^{k}\}), \tag{6}$$

$$\boldsymbol{\varphi}_{3,j}^{k+1}=\boldsymbol{\varphi}_{3,j}^{k}+\eta_{\boldsymbol{\varphi}}\nabla_{\boldsymbol{\varphi}_{3,j}}L_{p,3}(\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3^{k+1\prime},\{\boldsymbol{x}_{3,j}^{k+1\prime}\},\{\boldsymbol{\varphi}_{3,j}^{k}\}), \tag{7}$$

where $\eta_{\boldsymbol{z}}$ and $\eta_{\boldsymbol{\varphi}}$ represent the step-sizes. Then, master broadcasts the $\boldsymbol{z}_3^{k+1\prime}$ and $\boldsymbol{\varphi}_{3,j}^{k+1}$ to workers.

The results after K communication rounds are utilized as the estimate of $\phi_{\text{I}}(\boldsymbol{z}_1,\boldsymbol{z}_2')$, that is,

$$\phi_{\text{I}}(\boldsymbol{z}_1,\boldsymbol{z}_2')=\begin{bmatrix}\{\boldsymbol{x}_{3,j}^{0\ \prime}-\sum_{k=0}^{K-1}\eta_{\boldsymbol{x}}\nabla_{\boldsymbol{x}_{3,j}}L_{p,3}^{k}\}\\ \boldsymbol{z}_3^{0\prime}-\sum_{k=0}^{K-1}\eta_{\boldsymbol{z}}\nabla_{\boldsymbol{z}_3}L_{p,3}^{k}\end{bmatrix}, \tag{8}$$

where $L_{p,3}^{k}=L_{p,3}(\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3^{k\prime},\{\boldsymbol{x}_{3,j}^{k\ \prime}\},\{\boldsymbol{\varphi}_{3,j}^{k}\})$. Based on Eq. (8) and the definition of h_{I}, we have that,

$$\begin{aligned}&h_{\text{I}}(\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3)\\ &=||\begin{bmatrix}\{\boldsymbol{x}_{3,j}-\boldsymbol{x}_{3,j}^{0\ \prime}+\sum_{k=0}^{K-1}\eta_{\boldsymbol{x}}\nabla_{\boldsymbol{x}_{3,j}}L_{p,3}^{k}\}\\ \boldsymbol{z}_3-\boldsymbol{z}_3^{0\prime}+\sum_{k=0}^{K-1}\eta_{\boldsymbol{z}}\nabla_{\boldsymbol{z}_3}L_{p,3}^{k}\end{bmatrix}||^2.\end{aligned} \tag{9}$$

Inspired by polyhedral approximation method (Bertsekas 2015; Bürger, Notarstefano, and Allgöwer 2013), the I$^{\text{st}}$ **layer polytope**, which forms of a set of cutting planes (i.e., linear inequalities), is utilized to approximate the feasible region with respect to the constraint $h_{\text{I}}(\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3)\le\varepsilon_{\text{I}}$, which is a relaxed form of constraint $h_{\text{I}}(\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3)=0$ in Eq. (9), and $\varepsilon_{\text{I}}>0$ is a pre-set constant. Specifically, the I$^{\text{st}}$ layer polytope in $(t+1)^{\text{th}}$ iteration can be expressed as $P_{\text{I}}^{t}=\{\boldsymbol{a}_{1,l}^{\text{I}\ \top}\boldsymbol{z}_1+\boldsymbol{a}_{2,l}^{\text{I}\ \top}\boldsymbol{z}_2'+\boldsymbol{a}_{3,l}^{\text{I}\ \top}\boldsymbol{z}_3+\sum_{j=1}^{N}\boldsymbol{b}_{j,l}^{\text{I}\ \top}\boldsymbol{x}_{3,j}\le c_l^{\text{I}}, l=1,\cdots,|P_{\text{I}}^{t}|\}$, where $|P_{\text{I}}^{t}|$ denotes the number of cutting planes in I$^{\text{st}}$ layer polytope and $\boldsymbol{a}_{i,l}^{\text{I}},\boldsymbol{b}_{j,l}^{\text{I}},c_l^{\text{I}}$ are parameters in l^{th} cutting plane (I$^{\text{st}}$ layer μ-cut). Defining $\hat{h}_{\text{I},l}(\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3)=\boldsymbol{a}_{1,l}^{\text{I}\ \top}\boldsymbol{z}_1+\boldsymbol{a}_{2,l}^{\text{I}\ \top}\boldsymbol{z}_2'+\boldsymbol{a}_{3,l}^{\text{I}\ \top}\boldsymbol{z}_3+\sum_{j=1}^{N}\boldsymbol{b}_{j,l}^{\text{I}\ \top}\boldsymbol{x}_{3,j}$, the resulting

(bilevel) problem can be expressed as,

$$\begin{aligned}&\min \textstyle\sum_{j=1}^{N} f_{1,j}(\boldsymbol{x}_{1,j},\boldsymbol{x}_{2,j},\boldsymbol{x}_{3,j}) \text{ s.t.}\\&\boldsymbol{x}_{1,j}=\boldsymbol{z}_1, j=1,\cdots,N\\&\{\boldsymbol{x}_{2,j}\},\boldsymbol{z}_2=\underset{\{\boldsymbol{x}_{2,j}'\},\boldsymbol{z}_2'}{\arg\min} \textstyle\sum_{j=1}^{N} f_{2,j}(\boldsymbol{z}_1,\boldsymbol{x}_{2,j}',\boldsymbol{x}_{3,j}) \text{ s.t.}\\&\qquad\boldsymbol{x}_{2,j}'=\boldsymbol{z}_2', j=1,\cdots,N\\&\qquad\hat{h}_{\mathrm{I},l}(\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3)\le c_l^{\mathrm{I}}, l=1,\cdots,|P_{\mathrm{I}}^t|\\&\text{var.}\qquad\{\boldsymbol{x}_{1,j}\},\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3.\end{aligned}\tag{10}$$

$\mathrm{II}^{\mathrm{nd}}$ **layer Polyhedral Approximation:** Defining function $h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3) = ||\begin{bmatrix}\{\boldsymbol{x}_{2,j}\}\\ \boldsymbol{z}_2\end{bmatrix} - \phi_{\mathrm{II}}(\boldsymbol{z}_1,\boldsymbol{z}_3,\{\boldsymbol{x}_{3,j}\})||^2$, where $\phi_{\mathrm{II}}(\boldsymbol{z}_1,\boldsymbol{z}_3,\{\boldsymbol{x}_{3,j}\}) = \arg\min_{\{\boldsymbol{x}_{2,j}'\},\boldsymbol{z}_2'}\{\sum_{j=1}^{N} f_{2,j}(\boldsymbol{z}_1,\boldsymbol{x}_{2,j}',\boldsymbol{x}_{3,j}) : \boldsymbol{x}_{2,j}' = \boldsymbol{z}_2', \forall j, \boldsymbol{a}_{1,l}^{\mathrm{I}\top}\boldsymbol{z}_1 + \boldsymbol{a}_{2,l}^{\mathrm{I}\top}\boldsymbol{z}_2' + \boldsymbol{a}_{3,l}^{\mathrm{I}\top}\boldsymbol{z}_3 + \sum_{j=1}^{N}\boldsymbol{b}_{j,l}^{\mathrm{I}\top}\boldsymbol{x}_{3,j} \le c_l^{\mathrm{I}}, \forall l\}$. In Eq. (10), the lower-level optimization problem can be viewed as the constraint to the upper-level optimization problem (Sinha, Malo, and Deb 2017; Gould et al. 2016), i.e., $h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3) = 0$. Likewise, following (Jiao et al. 2022b), the results after K communication rounds between the master and workers are utilized as the estimate of $\phi_{\mathrm{II}}(\boldsymbol{z}_1,\boldsymbol{z}_3,\{\boldsymbol{x}_{3,j}\})$. Specifically, for the lower-level optimization problem in Eq. (10), the augmented Lagrangian function is given,

$$\begin{aligned}&L_{p,2}(\boldsymbol{z}_1,\boldsymbol{z}_2',\{\boldsymbol{x}_{2,j}'\},\{s_l\},\{\gamma_l\},\{\boldsymbol{\varphi}_{2,j}\},\boldsymbol{z}_3,\{\boldsymbol{x}_{3,j}\})\\&=\textstyle\sum_{j=1}^{N}(f_{2,j}(\boldsymbol{z}_1,\boldsymbol{x}_{2,j}',\boldsymbol{x}_{3,j})+\boldsymbol{\varphi}_{2,j}^{\top}(\boldsymbol{x}_{2,j}'-\boldsymbol{z}_2')\\&+\tfrac{\kappa_2}{2}||\boldsymbol{x}_{2,j}'-\boldsymbol{z}_2'||^2)+\textstyle\sum_{l=1}^{|P_{\mathrm{I}}^{t+1}|}\gamma_l(\hat{h}_{\mathrm{I},l}(\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3)\\&-c_l^{\mathrm{I}}+s_l)+\textstyle\sum_{l=1}^{|P_{\mathrm{I}}^{t+1}|}\tfrac{\rho_2}{2}||\hat{h}_{\mathrm{I},l}(\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2',\boldsymbol{z}_3)-c_l^{\mathrm{I}}+s_l||^2,\end{aligned}\tag{11}$$

where $\gamma_l\in\mathbb{R}^1$, $\boldsymbol{\varphi}_{2,j}\in\mathbb{R}^{d_2}$ are dual variables, $s_l\in\mathbb{R}_+^1, \forall l$ are the slack variables introduced in the inequality constraints, constants $\kappa_2>0$, $\rho_2>0$ are penalty parameters. The details of each communication round are presented in Appendix B in the supplementary material. After K communication rounds, we can obtain the estimate of $\phi_{\mathrm{II}}(\boldsymbol{z}_1,\boldsymbol{z}_3,\{\boldsymbol{x}_{3,j}\})$ and the corresponding h_{II} can be expressed as,

$$\begin{aligned}&h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3)\\&=||\begin{bmatrix}\{\boldsymbol{x}_{2,j}-\boldsymbol{x}_{2,j}^{0\,\prime}+\sum_{k=0}^{K-1}\eta_{\boldsymbol{x}}\nabla_{\boldsymbol{x}_{2,j}}L_{p,2}^k\}\\ \boldsymbol{z}_2-\boldsymbol{z}_2^{0\prime}+\sum_{k=0}^{K-1}\eta_{\boldsymbol{z}}\nabla_{\boldsymbol{z}_2}L_{p,2}^k\end{bmatrix}||^2,\end{aligned}\tag{12}$$

where $L_{p,2}^k$ is the simplified form of $L_{p,2}(\boldsymbol{z}_1,\boldsymbol{z}_2^{k\prime},\{\boldsymbol{x}_{2,j}^{k\,\prime}\},\{s_l^k\},\{\gamma_l^k\},\{\boldsymbol{\varphi}_{2,j}^k\},\boldsymbol{z}_3,\{\boldsymbol{x}_{3,j}\})$. Next, relaxing constraint $h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3)=0$ and utilizing $\mathrm{II}^{\mathrm{nd}}$ **layer polytope** to approximate the feasible region of relaxed constraint $h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3)\le\varepsilon_{\mathrm{II}}$. Specifically, the $\mathrm{II}^{\mathrm{nd}}$ layer polytope can be expressed as $P_{\mathrm{II}}^t=\{\sum_{i=1}^{3}\boldsymbol{a}_{i,l}^{\mathrm{II}\top}\boldsymbol{z}_i+\sum_{i=2}^{3}\sum_{j=1}^{N}\boldsymbol{b}_{i,j,l}^{\mathrm{II}\top}\boldsymbol{x}_{i,j}\le c_l^{\mathrm{II}}, l=1,\cdots,|P_{\mathrm{II}}^t|\}$ in $(t+1)^{\mathrm{th}}$ iteration, where $|P_{\mathrm{II}}^t|$ represents the number of cutting planes in P_{II}^t, and $\boldsymbol{a}_{i,l}^{\mathrm{II}},\boldsymbol{b}_{i,j,l}^{\mathrm{II}},c_l^{\mathrm{II}}$ are parameters in l^{th} cutting plane ($\mathrm{II}^{\mathrm{nd}}$ layer μ-cut). Thus, the resulting **hyper-polyhedral approximation problem** is,

$$\begin{aligned}&\min \textstyle\sum_{j=1}^{N} f_{1,j}(\boldsymbol{x}_{1,j},\boldsymbol{x}_{2,j},\boldsymbol{x}_{3,j}) \text{ s.t.}\\&\boldsymbol{x}_{1,j}=\boldsymbol{z}_1, j=1,\cdots,N\\&\textstyle\sum_{i=1}^{3}\boldsymbol{a}_{i,l}^{\mathrm{II}\top}\boldsymbol{z}_i+\sum_{i=2}^{3}\sum_{j=1}^{N}\boldsymbol{b}_{i,j,l}^{\mathrm{II}\top}\boldsymbol{x}_{i,j}\le c_l^{\mathrm{II}}, l=1,\cdots,|P_{\mathrm{II}}^t|\\&\text{var.}\qquad\{\boldsymbol{x}_{1,j}\},\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3.\end{aligned}\tag{13}$$

It is worth mentioning that solving the TLO problem is theoretically NP-hard (even solving the inner bilevel problem in TLO is NP-hard (Ben-Ayed and Blair 1990)). Thus, it's unlikely to design a polynomial-time algorithm for the distributed TLO problem unless P = NP (Arora and Barak 2009). In this work, the hyper-polyhedral approximation problem in Eq. (13) is a convex relaxation problem of the distributed TLO problem in Eq. (2), and the relaxation will be continuously tightened as μ-cuts are added. Detailed discussions are provided in Appendix D.

Asynchronous Federated Algorithm

The synchronous and asynchronous federated algorithms have different application scenarios (Su et al. 2022). The synchronous algorithm is preferred when the delay of each worker is not much different, and the asynchronous algorithm suits better when there are stragglers in the distributed system. In this work, an asynchronous algorithm is proposed to solve the trilevel optimization problem. Specifically, in the proposed asynchronous algorithm, we set the master updates its variables once it receives updates from $S(1\le S\le N)$ workers, i.e., active workers, at every iteration, and every worker has to communicate with the master at least once every τ iterations to alleviate the staleness issues (Zhang and Kwok 2014). It is worth mentioning that S can be flexibly adjusted based on whether there are stragglers, the proposed algorithm becomes synchronous when we set $S=N$, thus the proposed asynchronous algorithm is effective and flexible. First, the Lagrangian function of Eq. (13) can be expressed as,

$$\begin{aligned}&L_p(\{\boldsymbol{x}_{1,j}\},\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3,\{\lambda_l\},\{\boldsymbol{\theta}_j\})\\&=\textstyle\sum_{j=1}^{N}f_{1,j}(\boldsymbol{x}_{1,j},\boldsymbol{x}_{2,j},\boldsymbol{x}_{3,j})+\sum_{j=1}^{N}\boldsymbol{\theta}_j^{\top}(\boldsymbol{x}_{1,j}-\boldsymbol{z}_1)\\&+\textstyle\sum_{l=1}^{|P_{\mathrm{II}}^t|}\lambda_l(\sum_{i=1}^{3}\boldsymbol{a}_{i,l}^{\mathrm{II}\top}\boldsymbol{z}_i+\sum_{i=2}^{3}\sum_{j=1}^{N}\boldsymbol{b}_{i,j,l}^{\mathrm{II}\top}\boldsymbol{x}_{i,j}-c_l^{\mathrm{II}}),\end{aligned}\tag{14}$$

where $\lambda_l\in\mathbb{R}_+^1$, $\boldsymbol{\theta}_j\in\mathbb{R}^{d_1}$ are dual variables. Following (Xu et al. 2020; Jiao et al. 2022b), the regularized Lagrangian function is used to update variables as follows,

$$\begin{aligned}&\widehat{L}_p(\{\boldsymbol{x}_{1,j}\},\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3,\{\lambda_l\},\{\boldsymbol{\theta}_j\})\\&=L_p-\textstyle\sum_{l=1}^{|P_{\mathrm{II}}^t|}\frac{c_1^t}{2}||\lambda_l||^2-\sum_{j=1}^{N}\frac{c_2^t}{2}||\boldsymbol{\theta}_j||^2,\end{aligned}\tag{15}$$

where $L_p=L_p(\{\boldsymbol{x}_{1,j}\},\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3,\{\lambda_l\},\{\boldsymbol{\theta}_j\})$, and c_1^t, c_2^t are the regularization terms in $(t+1)^{\mathrm{th}}$ iteration. We set that $c_1^t = 1/\eta_\lambda(t+1)^{\frac{1}{4}} \ge \underline{c}_1$, $c_2^t = 1/\eta_{\boldsymbol{\theta}}(t+1)^{\frac{1}{4}} \ge \underline{c}_2$ are two nonnegative non-increasing sequences, where η_λ, $\eta_{\boldsymbol{\theta}}$, $\underline{c}_1$, $\underline{c}_2$ are constants, and $\underline{c}_1$, $\underline{c}_2$ meet that $0<\underline{c}_1<1/\eta_\lambda((4M\alpha_4/{\eta_\lambda}^2+4N\alpha_5/{\eta_{\boldsymbol{\theta}}}^2)1/\epsilon)^{\frac{1}{2}}$ and $0<\underline{c}_2<1/\eta_{\boldsymbol{\theta}}((4M\alpha_4/{\eta_\lambda}^2+4N\alpha_5/{\eta_{\boldsymbol{\theta}}}^2)1/\epsilon)^{\frac{1}{2}}$ (ϵ refers to the tolerance error, and α_4, α_5 are constants, which

will be introduced below). In $(t+1)^{\text{th}}$ master iteration, Q^{t+1} is utilized to denote the index set of active workers, and the proposed asynchronous algorithm proceeds as follows,

(1) *Active workers* update the local variables as follows,

$$\boldsymbol{x}_{i,j}^{t+1} = \begin{cases} \boldsymbol{x}_{i,j}^{t} - \eta_{\boldsymbol{x}_i} \nabla_{\boldsymbol{x}_{i,j}} \widehat{L}_p^{\hat{t}_j}, j \in Q^{t+1} \\ \boldsymbol{x}_{i,j}^{t}, j \notin Q^{t+1} \end{cases}, \forall i, \quad (16)$$

where $\eta_{\boldsymbol{x}_i} (\forall i = 1,2,3)$ denote the step-sizes, $\widehat{L}_p^{\hat{t}_j} = \widehat{L}_p(\{\boldsymbol{x}_{i,j}^{\hat{t}_j}\},\{\boldsymbol{z}_i^{\hat{t}_j}\},\{\lambda_l^{\hat{t}_j}\},\{\boldsymbol{\theta}_j^{\hat{t}_j}\})$ and $\hat{t}_j$ denotes the last iteration that worker j is active. Then, active workers (i.e., worker $j, j \in Q^{t+1}$) transmit the updated local variables, i.e., $\boldsymbol{x}_{i,j}^{t+1}, \forall i$ to the master.

(2) After receiving the updates from workers, the *master* updates the variables as follows,

$$\boldsymbol{z}_1^{t+1} = \boldsymbol{z}_1^t - \eta_{\boldsymbol{z}_1} \nabla_{\boldsymbol{z}_1} \widehat{L}_p(\{\boldsymbol{x}_{i,j}^{t+1}\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\}), \quad (17)$$

$$\boldsymbol{z}_2^{t+1} = \boldsymbol{z}_2^t - \eta_{\boldsymbol{z}_2} \nabla_{\boldsymbol{z}_2} \widehat{L}_p(\{\boldsymbol{x}_{i,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^t,\boldsymbol{z}_3^t,\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\}), \quad (18)$$

$$\boldsymbol{z}_3^{t+1} = \boldsymbol{z}_3^t - \eta_{\boldsymbol{z}_3} \nabla_{\boldsymbol{z}_3} \widehat{L}_p(\{\boldsymbol{x}_{i,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1},\boldsymbol{z}_3^t,\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\}), \quad (19)$$

$$\lambda_l^{t+1} = \mathcal{P}_\Lambda(\lambda_l^t + \eta_\lambda \nabla_{\lambda_l} \widehat{L}_p(\{\boldsymbol{x}_{i,j}^{t+1}\},\{\boldsymbol{z}_i^{t+1}\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\})), \quad (20)$$

$$\boldsymbol{\theta}_j^{t+1} = \mathcal{P}_\Theta(\boldsymbol{\theta}_j^t + \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}_j} \widehat{L}_p(\{\boldsymbol{x}_{i,j}^{t+1}\},\{\boldsymbol{z}_i^{t+1}\},\{\lambda_l^{t+1}\},\{\boldsymbol{\theta}_j^t\})), \quad (21)$$

where $\eta_{\boldsymbol{z}_1}, \eta_{\boldsymbol{z}_2}, \eta_{\boldsymbol{z}_3}, \eta_\lambda, \eta_{\boldsymbol{\theta}}$ denote the step-sizes, $\mathcal{P}_\Lambda$ and $\mathcal{P}_\Theta$ represent the projection onto sets $\Lambda = \{\lambda_l |\ 0 \le \lambda_l \le \sqrt{\alpha_4}\}$ and $\Theta = \{\boldsymbol{\theta}_j |\ ||\boldsymbol{\theta}_j||_\infty \le \sqrt{\alpha_5}/d_1\}$, where $\alpha_4 > 0$ and $\alpha_5 > 0$ are constants. Then, master broadcasts the updated variables, i.e., $\boldsymbol{z}_1^{t+1}, \boldsymbol{z}_2^{t+1}, \boldsymbol{z}_3^{t+1}, \{\lambda_l^{t+1}\}, \boldsymbol{\theta}_j$ to the active worker j. Details are summarized in Algorithm 1.

Refining Hyper-polyhedral Approximation

In this section, a novel μ-cut is proposed, which can be utilized for non-convex (μ-weakly convex) optimization problem and thus is more general than the traditional cutting plane designed for convex optimization (Jiao et al. 2022b; Franc, Sonnenburg, and Werner 2011). We demonstrate that the proposed μ-cuts are valid, i.e., the original feasible region is a subset of the polytope that forms of μ-cuts in Proposition 1 and 2. Every T_{pre} iteration, the μ-cuts will be updated to refine the hyper-polyhedral approximation when $t < T_1$, which can be divided into three steps: 1) generating new I^{st} layer μ-cut, 2) generating new II^{nd} layer μ-cut, 3) removing inactive μ-cuts.

Generating new I^{st} layer μ-cut: Following (Qian et al. 2019), we assume the variables are bounded, i.e., $||\boldsymbol{x}_{i,j}||^2 \le \alpha_i$, $||\boldsymbol{z}_i||^2 \le \alpha_i, i = 1,2,3$, and h_{I} is μ-weakly convex. It is demonstrated in Appendix E that h_{I} is μ-weakly convex in lots of cases. Following (Xie, Koyejo, and Gupta 2019; Davis and Drusvyatskiy 2019), the definition and first-order condition of μ-weakly convex function are given as follows.

Definition 1 ***(μ-weakly convex)*** *A differentiable function $f(\boldsymbol{x})$ is μ-weakly convex if function $g(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{\mu}{2}||\boldsymbol{x}||^2$ is convex.*

Algorithm 1: Asynchronous Federated Trilevel Learning

Initialization: master iteration $t = 0$, variables $\{\boldsymbol{x}_{1,j}^0\}$, $\{\boldsymbol{x}_{2,j}^0\}$, $\{\boldsymbol{x}_{3,j}^0\}$, $\boldsymbol{z}_1^0, \boldsymbol{z}_2^0, \boldsymbol{z}_3^0, \{\lambda_l^0\}, \{\boldsymbol{\theta}_j^0\}$.
repeat
  for *active worker* **do**
    updates variables $\boldsymbol{x}_{1,j}^{t+1}$, $\boldsymbol{x}_{2,j}^{t+1}$ and $\boldsymbol{x}_{3,j}^{t+1}$ by Eq. (16);
  end for
  active workers send updated local variables to master;
  for *master* **do**
    updates variables $\boldsymbol{z}_1^{t+1}, \boldsymbol{z}_2^{t+1}, \boldsymbol{z}_3^{t+1}, \{\lambda_l^{t+1}\}, \{\boldsymbol{\theta}_j^{t+1}\}$ by Eq. (17), (18), (19), (20) and (21);
  end for
  master broadcasts updated variables to active workers;
  if $(t+1)$ mod $T_{\text{pre}} == 0$ and $t < T_1$ **then**
    new I^{st} layer μ-cut cp_{I} is generated by Eq. (23) and added into I^{st} layer polytope;
    new II^{nd} layer μ-cut cp_{II} is generated by Eq. (24) and added into II^{nd} layer polytope;
    removing inactive I^{st}, II^{nd} layer μ-cuts by Eq. (25);
  end if
  $t = t + 1$;
until termination.

Definition 2 ***(First-order condition)*** *For any $\boldsymbol{x}$, $\boldsymbol{x}'$, a differentiable function $f(\boldsymbol{x})$ is μ-weakly convex if and only if the following inequality holds.*

$$f(\boldsymbol{x}) \ge f(\boldsymbol{x}') + \nabla f(\boldsymbol{x}')^\top (\boldsymbol{x} - \boldsymbol{x}') - \frac{\mu}{2}||\boldsymbol{x} - \boldsymbol{x}'||^2. \quad (22)$$

Combining the first-order condition of μ-weakly convex function with Cauchy-Schwarz inequality, a kind of new cutting plane, i.e., μ-**cut**, can be generated. Specifically, the new I^{st} layer μ-cut cp_{I} for I^{st} layer polytope can be expressed as:

$$\begin{bmatrix} \{\frac{\partial h_{\text{I}}(\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1\prime},\boldsymbol{z}_3^{t+1})}{\partial \boldsymbol{x}_{3,j}}\} \\ \frac{\partial h_{\text{I}}(\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1\prime},\boldsymbol{z}_3^{t+1})}{\partial \boldsymbol{z}_1} \\ \frac{\partial h_{\text{I}}(\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1\prime},\boldsymbol{z}_3^{t+1})}{\partial \boldsymbol{z}_2'} \\ \frac{\partial h_{\text{I}}(\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1\prime},\boldsymbol{z}_3^{t+1})}{\partial \boldsymbol{z}_3} \end{bmatrix}^\top \begin{bmatrix} \{\boldsymbol{x}_{3,j} - \boldsymbol{x}_{3,j}^{t+1}\} \\ \boldsymbol{z}_1 - \boldsymbol{z}_1^{t+1} \\ \boldsymbol{z}_2' - \boldsymbol{z}_2^{t+1\prime} \\ \boldsymbol{z}_3 - \boldsymbol{z}_3^{t+1} \end{bmatrix}$$
$$+ h_{\text{I}}(\{\boldsymbol{x}_{3,j}^{t+1}\}, \boldsymbol{z}_1^{t+1}, \boldsymbol{z}_2^{t+1\prime}, \boldsymbol{z}_3^{t+1}) \le \varepsilon_{\text{I}} + \mu((N+1)\alpha_1 + \alpha_2 + \alpha_3 + \sum_{j=1}^N ||\boldsymbol{x}_{3,j}^{t+1}||^2 + ||\boldsymbol{z}_1^{t+1}||^2 + ||\boldsymbol{z}_2^{t+1\prime}||^2 + ||\boldsymbol{z}_3^{t+1}||^2). \quad (23)$$

Proposition 1 *The feasible region of constraint $h_{\text{I}}(\{\boldsymbol{x}_{3,j}\}, \boldsymbol{z}_1, \boldsymbol{z}_2', \boldsymbol{z}_3) \le \varepsilon_{\text{I}}$ is a subset of the I^{st} layer polytope $P_{\text{I}}^t = \{\boldsymbol{a}_{1,l}^{\text{I}\top} \boldsymbol{z}_1 + \boldsymbol{a}_{2,l}^{\text{I}\top} \boldsymbol{z}_2' + \boldsymbol{a}_{3,l}^{\text{I}\top} \boldsymbol{z}_3 + \sum_{j=1}^N \boldsymbol{b}_{j,l}^{\text{I}\top} \boldsymbol{x}_{3,j} \le c_l^{\text{I}}, l = 1, \cdots, |P_{\text{I}}^t|\}$. In addition, P_{I}^t converges monotonically with the number of μ-cuts. The proof is given in Appendix C.*

Note that if h_{I} is convex, i.e., $\mu = 0$, the cutting plane will be generated as the same as that in (Franc, Sonnenburg, and Werner 2011; Jiao et al. 2022b), which is designed for convex optimization. Thus, the proposed μ-cut is more general than prior work in the literature. Consequently, the I^{st} layer polytope will be updated as $P_{\text{I}}^{t+1} = \text{Add}(P_{\text{I}}^t, cp_{\text{I}})$,

where $\mathrm{Add}(P_{\mathrm{I}}^t, cp_{\mathrm{I}})$ represents adding new μ-cut cp_{I} into the polytope P_{I}^t.

Generating new $\mathrm{II}^{\mathrm{nd}}$ layer μ-cut: Based on the updated I^{st} layer polytope, the $\mathrm{II}^{\mathrm{nd}}$ layer polytope will be updated. The new generated $\mathrm{II}^{\mathrm{nd}}$ layer μ-cut cp_{II} can be written as,

$$\begin{aligned}&\begin{bmatrix}\{\frac{\partial h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}^{t+1}\},\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1},\boldsymbol{z}_3^{t+1})}{\partial \boldsymbol{x}_{2,j}}\}\\ \{\frac{\partial h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}^{t+1}\},\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1},\boldsymbol{z}_3^{t+1})}{\partial \boldsymbol{x}_{3,j}}\}\\ \frac{\partial h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}^{t+1}\},\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1},\boldsymbol{z}_3^{t+1})}{\partial \boldsymbol{z}_1}\\ \frac{\partial h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}^{t+1}\},\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1},\boldsymbol{z}_3^{t+1})}{\partial \boldsymbol{z}_2}\\ \frac{\partial h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}^{t+1}\},\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1},\boldsymbol{z}_3^{t+1})}{\partial \boldsymbol{z}_3}\end{bmatrix}^{\top}\begin{bmatrix}\{\boldsymbol{x}_{2,j}-\boldsymbol{x}_{2,j}^{t+1}\}\\ \{\boldsymbol{x}_{3,j}-\boldsymbol{x}_{3,j}^{t+1}\}\\ \boldsymbol{z}_1-\boldsymbol{z}_1^{t+1}\\ \boldsymbol{z}_2-\boldsymbol{z}_2^{t+1}\\ \boldsymbol{z}_3-\boldsymbol{z}_3^{t+1}\end{bmatrix}\\ &+h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}^{t+1}\},\{\boldsymbol{x}_{3,j}^{t+1}\},\boldsymbol{z}_1^{t+1},\boldsymbol{z}_2^{t+1},\boldsymbol{z}_3^{t+1})\le\varepsilon_{\mathrm{II}}+\mu(\alpha_1\\ &+(N+1)(\alpha_2+\alpha_3)+\textstyle\sum_{i=2}^{3}\sum_{j=1}^{N}||\boldsymbol{x}_{i,j}^{t+1}||^2+\sum_{i=1}^{3}||\boldsymbol{z}_i^{t+1}||^2).\end{aligned}\tag{24}$$

Consequently, the $\mathrm{II}^{\mathrm{nd}}$ layer polytope will be updated as $P_{\mathrm{II}}^{t+1}=\mathrm{Add}(P_{\mathrm{II}}^t, cp_{\mathrm{II}})$.

Proposition 2 *The feasible region of constraint $h_{\mathrm{II}}(\{\boldsymbol{x}_{2,j}\},\{\boldsymbol{x}_{3,j}\},\{\boldsymbol{z}_i\})\le\varepsilon_{\mathrm{II}}$ is a subset of the $\mathrm{II}^{\mathrm{nd}}$ layer polytope $P_{\mathrm{II}}^t=\{\sum_{i=1}^{3}{\boldsymbol{a}_{i,l}^{\mathrm{II}}}^{\top}\boldsymbol{z}_i+\sum_{i=2}^{3}\sum_{j=1}^{N}{\boldsymbol{b}_{i,j,l}^{\mathrm{II}}}^{\top}\boldsymbol{x}_{i,j}\le c_l^{\mathrm{II}}, l=1,\cdots,|P_{\mathrm{II}}^t|\}$, and P_{II}^t converges monotonically with the number of μ-cuts. The proof is given in Appendix C.*

Removing inactive μ-cuts: Removing the inactive cutting planes can enhance the efficiency of the proposed algorithm (Yang et al. 2014; Jiao, Yang, and Song 2022). The inactive I^{st} and $\mathrm{II}^{\mathrm{nd}}$ layer μ-cuts will be removed, thus the corresponding I^{st} and $\mathrm{II}^{\mathrm{nd}}$ layer polytopes P_{I}^{t+1} and P_{II}^{t+1} will be updated as follows.

$$\begin{aligned}P_{\mathrm{I}}^{t+1}&=\begin{cases}\mathrm{Drop}(P_{\mathrm{I}}^{t+1}, cp_l^{\mathrm{I}}), \text{if } \gamma_l^K=0\\ P_{\mathrm{I}}^{t+1}, \text{otherwise}\end{cases},\\ P_{\mathrm{II}}^{t+1}&=\begin{cases}\mathrm{Drop}(P_{\mathrm{II}}^{t+1}, cp_l^{\mathrm{II}}), \text{if } \lambda_l^{t+1}=0\\ P_{\mathrm{II}}^{t+1}, \text{otherwise}\end{cases},\end{aligned}\tag{25}$$

where $\mathrm{Drop}(P, cp_l)$ represents that the l^{th} cutting plane cp_l is removed from polytope P.

Discussion

Definition 3 ***(Stationarity gap)*** *Following (Xu et al. 2020; Lu et al. 2020; Jiao, Yang, and Song 2022), the stationarity gap of our problem at t^{th} iteration is defined as:*

$$\nabla G^t=\begin{bmatrix}\{\nabla_{\boldsymbol{x}_{i,j}}L_p(\{\boldsymbol{x}_{i,j}^t\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\})\}\\ \{\nabla_{\boldsymbol{z}_i}L_p(\{\boldsymbol{x}_{i,j}^t\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\})\}\\ \{\nabla G_{\lambda_l}(\{\boldsymbol{x}_{i,j}^t\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\})\}\\ \{\nabla G_{\boldsymbol{\theta}_j}(\{\boldsymbol{x}_{i,j}^t\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\})\}\end{bmatrix},\tag{26}$$

where

$$\begin{aligned}&\nabla G_{\lambda_l}(\{\boldsymbol{x}_{i,j}^t\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\})\\ &=\tfrac{1}{\eta_\lambda}\left(\lambda_l^t-\mathcal{P}_\Lambda(\lambda_l^t+\eta_\lambda\nabla_{\lambda_l}L_p(\{\boldsymbol{x}_{i,j}^t\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\}))\right),\\ &\nabla G_{\boldsymbol{\theta}_j}(\{\boldsymbol{x}_{i,j}^t\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\})\\ &=\tfrac{1}{\eta_{\boldsymbol{\theta}}}\left(\boldsymbol{\theta}_j^t-\mathcal{P}_\Theta(\boldsymbol{\theta}_j^t+\eta_{\boldsymbol{\theta}}\nabla_{\boldsymbol{\theta}_j}L_p(\{\boldsymbol{x}_{i,j}^t\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\}))\right).\end{aligned}\tag{27}$$

Definition 4 ***(ϵ-stationary point)*** *If $||\nabla G^t||^2\le\epsilon$, $(\{\boldsymbol{x}_{i,j}^t\},\{\boldsymbol{z}_i^t\},\{\lambda_l^t\},\{\boldsymbol{\theta}_j^t\})$ is an ϵ-stationary point ($\epsilon\ge0$) of a differentiable function L_p. $T(\epsilon)$ is the first iteration index such that $||\nabla G^t||^2\le\epsilon$, i.e., $T(\epsilon)=\min\{t\mid||\nabla G^t||^2\le\epsilon\}$.*

Assumption 1 ***(Gradient Lipschitz)*** *Following (Ji, Yang, and Liang 2021), we assume that L_p has Lipschitz continuous gradients, i.e., for any $\boldsymbol{\omega},\boldsymbol{\omega}'$, we assume that there exists $L>0$ satisfying that,*

$$||\nabla L_p(\boldsymbol{\omega})-\nabla L_p(\boldsymbol{\omega}')||\le L||\boldsymbol{\omega}-\boldsymbol{\omega}'||.\tag{28}$$

Assumption 2 ***(Boundedness)*** *Following (Qian et al. 2019; Jiao et al. 2022b), we assume $||\boldsymbol{x}_{i,j}||^2\le\alpha_i$, $||\boldsymbol{z}_i||^2\le\alpha_i, i=1,2,3$. And we assume that before obtaining the ϵ-stationary point (i.e., $t\le T(\epsilon)-1$), the variables in master satisfy that $\sum_i||\boldsymbol{z}_i^{t+1}-\boldsymbol{z}_i^t||^2+\sum_l||\lambda_l^{t+1}-\lambda_l^t||^2\ge\vartheta$, where $\vartheta>0$ is a relative small constant. The change of the variables in master is upper bounded within τ iterations:*

$$||\boldsymbol{z}_i^t-\boldsymbol{z}_i^{t-k}||^2\le\tau k_1\vartheta, \textstyle\sum_l||\lambda_l^t-\lambda_l^{t-k}||^2\le\tau k_1\vartheta, 1\le k\le\tau,\tag{29}$$

where $k_1>0$ is a constant. Detailed discussions about Assumption 1 and 2 are provided in Appendix I.

Theorem 1 ***(Iteration Complexity)*** *Suppose Assumption 1 and 2 hold, we set the step-sizes as $\eta_{\boldsymbol{x}_i}=\eta_{\boldsymbol{z}_i}=\frac{2}{L+\eta_\lambda ML^2+\eta_{\boldsymbol{\theta}}NL^2+8(\frac{M\gamma L^2}{\eta_\lambda\varsigma_1^2}+\frac{N\gamma L^2}{\eta_{\boldsymbol{\theta}}\varsigma_2^2})}, \forall i$, $\eta_{\boldsymbol{\theta}}\le\frac{2}{L+2c_2^0}$ and $\eta_\lambda<\min\{\frac{2}{L+2c_1^0},\frac{1}{30\tau k_1NL^2}\}$. For a given ϵ, we have:*

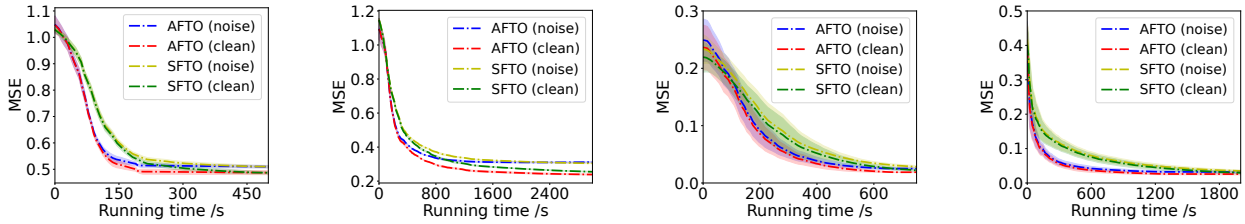
$$\begin{aligned}T(\epsilon)\sim\mathcal{O}(\max\{&(\tfrac{4M\alpha_4}{\eta_\lambda{}^2}+\tfrac{4N\alpha_5}{\eta_{\boldsymbol{\theta}}{}^2})^2\tfrac{1}{\epsilon^2},\\ &(\tfrac{4(d_9+\frac{\eta_{\boldsymbol{\theta}}(N-S)L^2}{2})(\bar{d}+k_d\tau(\tau-1))d_8}{\epsilon}+(T_1+2)^{\frac{1}{2}})^2\}),\end{aligned}\tag{30}$$

where α_4, α_5, γ, k_d, T_1, M, N, S, τ, $\bar{d}$, d_8 and d_9 are constants. The detailed proof is given in Appendix F. Moreover, the influence of parameters (e.g., T_1,τ,N,S) in iteration complexity is discussed in Appendix F in the supplementary material.

Theorem 2 ***(Communication Complexity)*** *The overall communication complexity of the proposed algorithm can be divided into complexity at every iteration and complexity of updating μ-cuts, which can be expressed as $\mathcal{O}(\sum_{t=1}^{T(\epsilon)}C_1^t+C_2)$, where $C_1^t=32S(2\sum_{i=1}^{3}d_i+d_1+|P_{\mathrm{II}}^t|)$, $C_2=32\sum_{t\in\mathcal{Q}}(NK(3\sum_{i=2}^{3}d_i+2|P_{\mathrm{II}}^t|)+N|P_{\mathrm{II}}^t|(2\sum_{i=2}^{3}d_i+d_1+1))$, and set $\mathcal{Q}=\{T_{\mathrm{pre}},\cdots,\lfloor\frac{T_1}{T_{\mathrm{pre}}}\rfloor\cdot T_{\mathrm{pre}}\}$. Detailed proof is provided in Appendix J in supplementary material.*

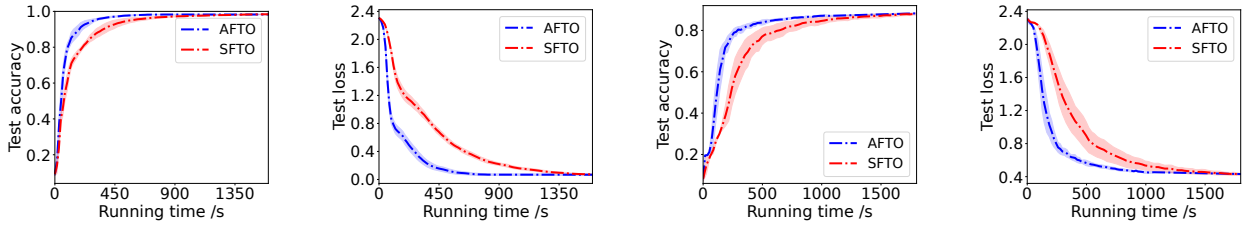
Experiment

In the experiment, two distributed trilevel optimization tasks are employed to assess the performance of the proposed method. In the distributed robust hyperparameter optimization, experiments are carried out on the regression tasks following (Sato, Tanaka, and Takeda 2021), and in distributed domain adaptation for pretraining & finetuning, the multiple domain digits recognition task in (Qian et al. 2019; Wang et al. 2021) is considered. The details of the experimental setting are summarized in Table 1 and Appendix



(a) Diabetes (b) Boston (c) Red-wine (d) White-wine

Figure 1: MSE of clean test data and test data with Gaussian noise on (a) Diabetes, (b) Boston, (c) Red-wine quality, and (d) White-wine quality datasets. All experiments are repeated five times, and the shaded areas represent the standard deviation.



(a) Test accuracy (b) Test loss (c) Test accuracy (d) Test loss

Figure 2: (a) Test accuracy and (b) test loss vs running time when SVHN is utilized to pretrain the model. (c) Test accuracy and (d) test loss vs running time when MNIST is utilized to pretrain the model. All experiments are repeated five times.

H. More experimental results are reported in Appendix G. To further show the superior performance of the proposed method, experimental results of comparisons between the non-distributed version of the proposed method with existing state-of-the-art TLO methods (Sato, Tanaka, and Takeda 2021; Choe et al. 2022) on three TLO tasks are shown in Appendix A in the supplementary material.

Distributed Robust Hyperparameter Optimization

The robust hyperparameter optimization (Sato, Tanaka, and Takeda 2021) aims to train a machine learning model that is robust against the noise in input data, which is inspired by bilevel hyperparameter optimization (Chen et al. 2022b) and adversarial training (Han, Shi, and Huang 2023; Zhang et al. 2022). And we consider the following distributed robust hyperparameter optimization problem,

$$\begin{aligned}
&\min \textstyle\sum_j \frac{1}{|D_j^{\mathrm{val}}|}||y_j^{\mathrm{val}} - f(X_j^{\mathrm{val}}; \boldsymbol{w})||^2 \text{ s.t.} \\
&\boldsymbol{p} = \arg\max_{\boldsymbol{p}'} \textstyle\sum_j (\frac{1}{|D_j^{\mathrm{tr}}|}||y_j^{\mathrm{tr}} - f(X_j^{\mathrm{tr}} + p_j'; \boldsymbol{w})||^2 - c||p_j'||^2) \text{ s.t.} \\
&\boldsymbol{w} = \arg\min_{\boldsymbol{w}'} \textstyle\sum_j (\frac{1}{|D_j^{\mathrm{tr}}|}||y_j^{\mathrm{tr}} - f(X_j^{\mathrm{tr}} + p_j'; \boldsymbol{w}')||^2 + e^{\boldsymbol{\varphi}}||\boldsymbol{w}'||_{1*}) \\
&\text{var.} \qquad \boldsymbol{\varphi}, \boldsymbol{p}, \boldsymbol{w},
\end{aligned} \tag{31}$$

where $\boldsymbol{\varphi}$, $\boldsymbol{w}$ and $\boldsymbol{p}$ respectively denote the regularization parameter, model parameter, and adversarial noise, $\boldsymbol{p}' = [p_1', \cdots, p_N']$, N is the number of workers. f denotes the output of a MLP, c denotes the penalty for the adversarial noise, and $||\cdot||_{1*}$ is a smoothed l_1-norm (Saheya, Nguyen, and Chen 2019). $X_j^{\mathrm{val}}, y_j^{\mathrm{val}}, |D_j^{\mathrm{val}}|$, $X_j^{\mathrm{tr}}, y_j^{\mathrm{tr}}, |D_j^{\mathrm{tr}}|$ respectively denote the data, label and the number of data of the validation and training datasets on local worker j. Following (Sato, Tanaka, and Takeda 2021), the experiments are carried out on the regression tasks with the following datasets: Diabetes (Dua, Graff et al. 2017), Boston (Harrison Jr and Rubinfeld 1978), Red-wine and White-wine quality (Cortez et al. 2009) datasets. We summarize the experimental setting on each dataset in Table 1. To show the performance of the proposed AFTO, we report the mean squared error (MSE) of clean test data and test data with Gaussian noise vs running time of the AFTO and SFTO (Synchronous Federated Trilevel Optimization) in Figure 1. It is seen that the proposed AFTO can effectively solve the TLO problem in a distributed manner and converges much faster than SFTO since the master can update its variables once it receives updates from a subset of workers instead of all workers in AFTO. Furthermore, we compare the proposed method with the state-of-the-art distributed bilevel optimization methods ADBO (Jiao et al. 2022b) and FEDNEST (Tarzanagh et al. 2022). It is shown in Table 2 that the proposed AFTO can achieve superior performance, which demonstrates the effectiveness of the proposed method.

	N	S	Stragglers	τ
Diabetes	4	3	1	10
Boston	4	3	1	10
Red-wine	4	3	1	10
White-wine	6	4	1	10
SVHN (finetune)	4	3	1	5
SVHN (pretrain)	6	3	2	15

Table 1: Experimental setting in distributed robust hyperparameter optimization and distributed domain adaptation.

Method	Diabetes	Boston	Red-wine	White-wine
FEDNEST	0.5293 ± 0.0229	0.3509 ± 0.0177	0.0339 ± 0.0014	0.0268 ± 0.0010
ADBO	0.5284 ± 0.0074	0.3243 ± 0.0046	0.0336 ± 0.0018	0.0277 ± 0.0013
AFTO	**0.5124 ± 0.0068**	**0.3130 ± 0.0037**	**0.0321 ± 0.0026**	**0.0248 ± 0.0021**

Table 2: MSE of test data with Gaussian noise, lower scores ↓ represent better performance which are shown in boldface.

Distributed Domain Adaptation

Pretraining/finetuning paradigms are increasingly adopted recently in self-supervised learning (He et al. 2020). In (Raghu et al. 2021), a domain adaptation strategy is proposed, which combines data reweighting with a pretraining/finetuning framework to automatically decrease/increase the weight of pretraining samples that cause negative/positive transfer, and can be formulated as trilevel optimization (Choe et al. 2022). The corresponding distributed trilevel optimization problem is given as follows,

$$\begin{aligned}
&\min \textstyle\sum_j L_{FT,j}(\boldsymbol{\varphi}, \boldsymbol{v}, \boldsymbol{w}) \text{ s.t.} \\
&\boldsymbol{v} = \arg\min_{\boldsymbol{v}'} \textstyle\sum_j \left(L_{FT,j}(\boldsymbol{\varphi}, \boldsymbol{v}', \boldsymbol{w}) + \lambda ||\boldsymbol{v}' - \boldsymbol{w}||^2\right) \text{ s.t.} \\
&\boldsymbol{w} = \arg\min_{\boldsymbol{w}'} \textstyle\sum_j \frac{1}{\mathcal{D}_j} \sum_{x_{i,j} \in \mathcal{D}_j} \mathcal{R}(x_{i,j}, \boldsymbol{\varphi}) \cdot L^i_{PT,j}(\boldsymbol{\varphi}, \boldsymbol{v}', \boldsymbol{w}') \\
&\text{var.} \qquad \boldsymbol{\varphi}, \boldsymbol{v}, \boldsymbol{w},
\end{aligned} \tag{32}$$

where $\boldsymbol{\varphi}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ respectively denote the parameters for pretraining, finetuning, and reweighting networks. $x_{i,j}$ and $L^i_{PT,j}$ represent the i^{th} pretraining sample and loss in worker j, $L_{FT,j}$ represents the finetuning loss in worker j. $\mathcal{R}(x_{i,j}, \boldsymbol{\varphi})$ denotes the importance of pretraining sample $x_{i,j}$, and λ is the proximal regularization parameter. To evaluate the performance of the proposed method, the multiple domain digits recognition task in (Qian et al. 2019; Wang et al. 2021) is considered. There are two benchmark datasets for this task: MNIST (LeCun et al. 1998) and SVHN (Netzer et al. 2011). In the experiments, we utilize the same image resize strategy as in (Qian et al. 2019) to make the format consistent, and LeNet-5 is used for all pretraining/finetuning/reweighting networks. We summarize the experimental setting in Table 1 and Appendix H. Following (Ji, Yang, and Liang 2021), we utilize the test accuracy/test loss vs running time to evaluate the proposed AFTO. It is seen from Figure 2 that the proposed AFTO can effectively solve the distributed trilevel optimization problem and exhibits superior performance, which achieves a faster convergence rate than SFTO with a maximum acceleration of approximately 80%.

Conclusion

Existing trilevel learning works focus on the non-distributed setting which may lead to data privacy risks, and do not provide the non-asymptotic analysis. To this end, we propose an asynchronous federated trilevel optimization method for TLO problems. To our best knowledge, this work takes an initial step that aims to solve the TLO problems in an asynchronous federated manner. The proposed μ-cuts are utilized to construct the hyper-polyhedral approximation for TLO problems, and it is demonstrated that they are applicable to a wide range of non-convex functions that meet the μ-weakly convex assumption. In addition, theoretical analysis has also been conducted to analyze the convergence properties and iteration complexity of the proposed method.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 12371519 and 61771013; in part by the Fundamental Research Funds for the Central Universities of China; in part by the Fundamental Research Funds of Shanghai Jiading District; in part by the National Natural Science Foundation of China (Project 62271434), Shenzhen Science and Technology Program (Project JCYJ20210324120011032), Guangdong Basic and Applied Basic Research Foundation (Project 2021B1515120008), Shenzhen Key Lab of Crowd Intelligence Empowered Low-Carbon Energy Network (No. ZDSYS20220606100601002), and the Shenzhen Institute of Artificial Intelligence and Robotics for Society.

References

Arora, S.; and Barak, B. 2009. *Computational complexity: a modern approach*. Cambridge University Press.

Assran, M.; Aytekin, A.; Feyzmahdavian, H. R.; Johansson, M.; and Rabbat, M. G. 2020. Advances in asynchronous parallel and distributed optimization. *Proceedings of the IEEE*, 108(11): 2013–2031.

Avraamidou, S. 2018. Mixed-integer multi-level optimization through multi-parametric programming.

Ben-Ayed, O.; and Blair, C. E. 1990. Computational difficulties of bilevel linear programming. *Operations Research*, 38(3): 556–560.

Bertsekas, D. 2015. *Convex optimization algorithms*. Athena Scientific.

Bertsekas, D. P.; and Yu, H. 2011. A unifying polyhedral approximation framework for convex optimization. *SIAM Journal on Optimization*, 21(1): 333–360.

Blair, C. 1992. The computational complexity of multi-level linear programs. *Annals of Operations Research*, 34.

Bürger, M.; Notarstefano, G.; and Allgöwer, F. 2013. A polyhedral approximation framework for convex and robust distributed optimization. *IEEE Transactions on Automatic Control*, 59(2): 384–395.

Chen, S.; Feng, S.; Guo, Z.; and Yang, Z. 2022a. Trilevel optimization model for competitive pricing of electric vehicle charging station considering distribution locational marginal price. *IEEE Transactions on Smart Grid*, 13(6): 4716–4729.

Chen, T.; Sun, Y.; Xiao, Q.; and Yin, W. 2022b. A single-timescale method for stochastic bilevel optimization. In *International Conference on Artificial Intelligence and Statistics*, 2466–2488. PMLR.

Choe, S. K.; Neiswanger, W.; Xie, P.; and Xing, E. 2022. Betty: An automatic differentiation library for multilevel optimization. *arXiv preprint arXiv:2207.02849*.

Cortez, P.; Cerdeira, A.; Almeida, F.; Matos, T.; and Reis, J. 2009. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4): 547–553.

Davis, D.; and Drusvyatskiy, D. 2019. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1): 207–239.

Dua, D.; Graff, C.; et al. 2017. UCI machine learning repository.

Franc, V.; Sonnenburg, S.; and Werner, T. 2011. Cutting plane methods in machine learning. *Optimization for Machine Learning*, 185–218.

Franceschi, L.; Frasconi, P.; Salzo, S.; Grazzi, R.; and Pontil, M. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, 1568–1577. PMLR.

Garg, B.; Zhang, L.; Sridhara, P.; Hosseini, R.; Xing, E.; and Xie, P. 2022. Learning from mistakes–a framework for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 10184–10192.

Gould, S.; Fernando, B.; Cherian, A.; Anderson, P.; Cruz, R. S.; and Guo, E. 2016. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*.

Guo, M.; Yang, Y.; Xu, R.; Liu, Z.; and Lin, D. 2020. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 631–640.

Han, P.; Shi, X.; and Huang, J. 2023. FedAL: Black-Box Federated Knowledge Distillation Enabled by Adversarial Learning. *arXiv preprint arXiv:2311.16584*.

Han, P.; Wang, S.; and Leung, K. K. 2020. Adaptive gradient sparsification for efficient federated learning: An online learning approach. In *2020 IEEE 40th international conference on distributed computing systems (ICDCS)*, 300–310. IEEE.

Harrison Jr, D.; and Rubinfeld, D. L. 1978. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1): 81–102.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9729–9738.

Ji, K.; Yang, J.; and Liang, Y. 2021. Bilevel optimization: Convergence analysis and enhanced design. In *International Conference on Machine Learning*, 4882–4892. PMLR.

Jiao, Y.; Yang, K.; and Song, D. 2022. Distributed distributionally robust optimization with non-convex objectives. *Advances in neural information processing systems*, 35: 7987–7999.

Jiao, Y.; Yang, K.; Song, D.; and Tao, D. 2022a. TimeAutoAD: Autonomous Anomaly Detection With Self-Supervised Contrastive Loss for Multivariate Time Series. *IEEE Transactions on Network Science and Engineering*, 9(3): 1604–1619.

Jiao, Y.; Yang, K.; Wu, T.; Song, D.; and Jian, C. 2022b. Asynchronous Distributed Bilevel Optimization. In *The Eleventh International Conference on Learning Representations*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Liu, R.; Liu, Y.; Zeng, S.; and Zhang, J. 2021. Towards gradient-based bilevel optimization with non-convex followers and beyond. *Advances in Neural Information Processing Systems*, 34: 8662–8675.

Lu, S.; Tsaknakis, I.; Hong, M.; and Chen, Y. 2020. Hybrid block successive approximation for one-sided non-convex min-max problems: algorithms and applications. *IEEE Transactions on Signal Processing*, 68: 3676–3691.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning.

Qian, Q.; Zhu, S.; Tang, J.; Jin, R.; Sun, B.; and Li, H. 2019. Robust optimization over multiple domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4739–4746.

Raghu, A.; Lorraine, J.; Kornblith, S.; McDermott, M.; and Duvenaud, D. K. 2021. Meta-learning to improve pre-training. *Advances in Neural Information Processing Systems*, 34: 23231–23244.

Saheya, B.; Nguyen, C. T.; and Chen, J.-S. 2019. Neural network based on systematically generated smoothing functions for absolute value equation. *Journal of Applied Mathematics and Computing*, 61(1): 533–558.

Sato, R.; Tanaka, M.; and Takeda, A. 2021. A Gradient Method for Multilevel Optimization. *Advances in Neural Information Processing Systems*, 34: 7522–7533.

Sinha, A.; Malo, P.; and Deb, K. 2017. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2): 276–295.

Su, W.; Zhang, Y.; Cai, Y.; Ren, K.; Wang, P.; Yi, H.; Song, Y.; Chen, J.; Deng, H.; Xu, J.; et al. 2022. GBA: A Tuning-free Approach to Switch between Synchronous and Asynchronous Training for Recommendation Models. *Advances in Neural Information Processing Systems*, 35: 29525–29537.

Subramanya, T.; and Riggio, R. 2021. Centralized and federated learning for predictive VNF autoscaling in multi-domain 5G networks and beyond. *IEEE Transactions on Network and Service Management*, 18(1): 63–78.

Tarzanagh, D. A.; Li, M.; Thrampoulidis, C.; and Oymak, S. 2022. Fednest: Federated bilevel, minimax, and compositional optimization. In *International Conference on Machine Learning*, 21146–21179. PMLR.

Tawarmalani, M.; and Sahinidis, N. V. 2005. A polyhedral branch-and-cut approach to global optimization. *Mathematical programming*, 103(2): 225–249.

Trombettoni, G.; Araya, I.; Neveu, B.; and Chabert, G. 2011. Inner regions and interval linearizations for global optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 99–104.

Wang, J.; Chen, J.; Lin, J.; Sigal, L.; and de Silva, C. W. 2021. Discriminative feature alignment: Improving transferability of unsupervised domain adaptation by Gaussian-guided latent alignment. *Pattern Recognition*, 116: 107943.

Xie, C.; Koyejo, S.; and Gupta, I. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*.

Xu, Z.; Zhang, H.; Xu, Y.; and Lan, G. 2020. A unified single-loop alternating gradient projection algorithm for nonconvex-concave and convex-nonconcave minimax problems. *arXiv preprint arXiv:2006.02032*.

Yang, K.; Huang, J.; Wu, Y.; Wang, X.; and Chiang, M. 2014. Distributed robust optimization (DRO), part I: Framework and example. *Optimization and Engineering*, 15(1): 35–67.

Yang, K.; Wu, Y.; Huang, J.; Wang, X.; and Verdú, S. 2008. Distributed robust optimization for communication networks. In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, 1157–1165. IEEE.

Zhang, R.; and Kwok, J. 2014. Asynchronous distributed ADMM for consensus optimization. In *International conference on machine learning*, 1701–1709. PMLR.

Zhang, Y.; Zhang, G.; Khanduri, P.; Hong, M.; Chang, S.; and Liu, S. 2022. Revisiting and advancing fast adversarial training through the lens of bi-level optimization. In *International Conference on Machine Learning*, 26693–26712. PMLR.