OPT2023: 15th Annual Workshop on Optimization for Machine Learning

A Predicting Clipping Asynchronous Stochastic Gradient Descent Method in Distributed Learning

Haoxiang Wang WHX22@MAILS.TSINGHUA.EDU.CN
Department of Automation, Tsinghua University
Zhanhong Jiang ZHJIANG@IASTATE.EDU
Translational AI Center, Iowa State University
Chao Liu CLIU5@TSINGHUA.EDU.CN
Department of Energy and Power Engineering, Tsinghua University
Soumik Sarkar SOUMIKS@IASTATE.EDU
Department of Mechanical Engineering, Iowa State University
Dongxiang Jiang JIANGDX@TSINGHUA.EDU.CN
Department of Energy and Power Engineering, Tsinghua University
Young M. Lee YOUNG.M.LEE@JCI.COM
Johnson Controls

Abstract

In this paper, we propose a new algorithm, termed **P**redicting **C**lipping **A**synchronous **S**tochastic **G**radient **D**escent (aka, PC-ASGD) to address the issue of staleness and time delay in asynchronous distributed learning settings. Specifically, PC-ASGD has two steps - the *predicting step* leverages the gradient prediction using Taylor expansion to reduce the staleness of the outdated weights while the *clipping step* selectively drops the outdated weights to alleviate their negative effects. A tradeoff parameter is introduced to balance the effects between these two steps. We theoretically present the convergence rate considering the effects of delay of the proposed algorithm with constant step size when the smooth objective functions are nonconvex. For empirical validation, we demonstrate the performance of the algorithm with two deep neural network architectures on two benchmark datasets.

1. Introduction

The availability of large datasets and powerful computing led to the emergence of deep learning that is revolutionizing many application sectors from the internet industry and healthcare to transportation and energy [7, 8, 13, 20]. As the applications are scaling up, the learning process of large deep learning models is looking to leverage emerging resources such as edge computing and distributed data centers privacy preserving. In this regard, distributed deep learning algorithms are being explored by the community, which leverage synchronous and asynchronous computations with multiple computing agents that exchange information over communication networks [3, 11, 16]. While the computing resources within a local cluster can operate in a (loosely) synchronous manner, multiple (geographically distributed) clusters may need to operate in an asynchronous manner. Furthermore, communications among the computing resources may not be reliable and prone to delay.

To address the aforementioned issue, Asynchronous Stochastic Gradient Descent (ASGD) algorithm in the master-slave manner has been proposed [4], which could tolerate the delay in communication. Later works [1, 6, 17, 25] extend ASGD to more realistic scenarios and implement the algorithms with a central server and other parallel workers. Typically, since asynchronous algorithms suffer from stale gradients, researchers have proposed algorithms such as DC-ASGD [24], adopting the concept of delay compensation to reduce the impact of staleness and improve the performance of ASGD. While for peer-to-peer architecture, [11] proposes an algorithm termed AD-PSGD (decentralized ASGD algorithm, aka D-ASGD) that deals with the problem of the stale parameter exchange and presents theoretical analysis under bounded delay. More recently, [18] proposes the DC-s3gd algorithm to enable large-scale decentralized neural network training with the consideration of delay. Asynchronous version of stochastic gradient push (AGP) [2] is developed and shown to be more robust to failing or stalling agents, though only applicable to the strongly convex objectives. To further advance this area, the most recent schemes such as Praque [14] adopting a partial all-reduce communication primitive, DSGD-AAU [22] utilizing an adaptive asynchronous updates, and DGD-ATC [21] using the Adapt-then-Combine technique, are presented, but most of them are limited to only (strongly) convex cases.

The contributions of this work are specifically as follows. PC-ASGD is proposed to tackle the convergence issues due to the varying communication delays, consisting of the predicting and clipping steps with an introduced tradeoff parameter. We show that with a proper constant step size, PC-ASGD converges to the *neighborhood* of the optimal solution at a sublinear rate for nonconvex functions (See Table 1 for comparison). PC-ASGD is deployed on distributed GPUs with two datasets CIFAR-10 and CIFAR-100 by using PreResNet110 and DenseNet architectures for validation.

Table 1: Comparisons between asynchronous algorithms

Methods	f	∇f	Delay Ass.	Rate	G.C.	A.S.
D-ASGD [11]	Non-convex	Lip.&Bou.	Bou.	$\mathcal{O}(\frac{1}{\sqrt{T}})$	✗	✗
DC-ASGD [24]	Non-convex	Lip.	Bou.	$\mathcal{O}(\frac{1}{\sqrt{T}})$	✓	✗
DC-s3dg [18]	Non-convex	Lip.	Unbou.	N/A	✓	✗
AGP [2]	Strongly-convex	Lip.	Bou.	$\mathcal{O}(\frac{1}{T}+\frac{1}{T^{\zeta}}+\frac{1}{T^{1-\zeta}})$	✗	✗
DSGD-AAU [22]	Non-convex	Lip.	Bou.	$\mathcal{O}(\frac{1}{\sqrt{T}})$	✗	✗
DGD-ATC [21]	Strongly-convex	Lip.	Unbou.	$\mathcal{O}(\rho^{T})$	✗	✗
PC-ASGD (This paper)	Non-convex	Lip.	Bou.	$\mathcal{O}(\frac{1}{\sqrt{T}})$	✓	✓

Lip.& Bou.: Lipschitz continuous and bounded. Delay Ass.: Delay Assumption. Unbou.: Unbounded. T: Total iterations. G.C.: Gradient Compensation. A.S.: Alternant Step, $\rho \in (0,1)$ is a positive constant. $\zeta \in (0,1)$.

2. Formulation and Preliminaries

Consider N agents in a networked system such that their interactions are driven by a graph $\mathcal{G}$, where $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, 2, .., N\}$ indicates the node or agent set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. Throughout the paper, we assume that the graph is undirected and connected. If agent j is in the neighborhood of agent i, they can communicate with each other. Thus, we define the neighborhood for any agent i as $Nb(i) := \{j \in \mathcal{V} | (i,j) \in \mathcal{E} \text{ or } j = i\}$. Rather than considering synchronization and asynchronization separately, this paper considers both scenarios together by defining the following terminology.

Definition 1 *At a time step t, an agent j is called a* ***reliable neighbor*** *of the agent i if agent i has the state information (the model parameters in this study) of agent j up to $t-1$. Similarly, an agent j is called an* ***unreliable neighbor*** *of the agent i if agent i has the state information of agent j only up to $t-\tau$, where τ is the so-called delay and $1 < \tau < \infty$.*

Denote by x the state information. Thus, inside the neighborhood of an agent, there are reliable and unreliable neighbors respectively. This work aims at studying how to effectively tackle issues such as negative impacts that delays may bring on the performance. We define a set for reliable neighbors of agent i as: $\mathcal{R} := \{j \in Nb(i) \mid \boldsymbol{Pr}(x^j = x^j_{t-1}|t) = 1\}$, implying that agent j has the state information x up to the time $t-1$, i.e., x^j_{t-1}. We can directly have the set for unreliable neighbors such that $\mathcal{R}^c = Nb \setminus \mathcal{R}$ [1].

We consider the decentralized empirical risk minimization problems as follows:

$$\min \ F(\mathbf{x}) := \sum_{i=1}^{N} \sum_{s \in \mathcal{D}_i} f_i(x^i; s) \tag{1}$$

where $\mathbf{x} = [x^1; x^2; ...; x^N]$, $x^i \in \mathbb{R}^d$ is the local model parameters, $\mathcal{D}_i$ is a local data set uniquely known by agent i, $f_i : \mathbb{R}^d \to \mathbb{R}$ is the incurred local loss of agent i given a sample s (we will drop this for simplicity). Based on the above formulation, we then assume everywhere that our objective function is bounded from below and denote the minimum by $F^* := F(\mathbf{x}^*)$ where $\mathbf{x}^* :=$ argmin $F(\mathbf{x})$. Hence $F^* > -\infty$. Moreover, all vector norms refer to the Euclidean norm. Some necessary assumptions are defined in the sequel.

Assumption 1 *1) Each objective function f_i is assumed to satisfy the following conditions: f_i is $\gamma_i -$ smooth and proper (not everywhere infinite) and coercive; 2) A mixing matrix $\underline{W} \in \mathbb{R}^{N\times N}$ satisfies a) $\mathbf{1}^\top \underline{W} = \mathbf{1}^\top, \underline{W}\mathbf{1}^\top = \mathbf{1}^\top$; b) Null$\{I - \underline{W}\}$ = Span$\{\mathbf{1}\}$ and $I \succeq \underline{W} \succ 0$; 3) The stochastic gradient of F at any $\mathbf{x}$ is denoted by $\mathbf{g}(\mathbf{x})$, such that a) $\mathbf{g}(\mathbf{x})$ is the unbiased estimate of gradient $\nabla F(\mathbf{x})$; b) The variance is uniformly bounded by σ^2, i.e.,$\mathbb{E}[\|\mathbf{g}(\mathbf{x}) - \nabla F(\mathbf{x})\|^2] \leq \sigma^2$; c) The second moment of $\mathbf{g}(\mathbf{x})$ is bounded, i.e., $\mathbb{E}[\|\mathbf{g}(\mathbf{x})\|^2] \leq G^2$.*

3. Algorithm and Main Result

We present the specific update law for our proposed method, PC-ASGD. In Algorithm 1, for the predicting step (line 6), any agent k that is unreliable has delay when communicating its weights with agent i. To compensate for the delay, we adopt the Taylor expansion to approximate the gradient for each time step. The predicted gradient (or delay compensated gradient) is denoted by $g_k^{dc,r}(x^k_{t-\tau})$ (See Appendix for its detailed derivation). For agent k, at t-th time step, since it did not get updated over the past τ time steps, it is known that $x^k_t := x^k_{t-\tau}$. By abuse of notation, we use $g_k^{dc,r}(x^k_{t-\tau})$ instead of $g_k^{dc,r}(x^k_t)$ for the predicted gradient. Additionally, the term $(x^i_{t-\tau+r} - x^i_{t-\tau})$ is from agent i due to the outdated information of agent k, which intuitively implies that the compensation is driven by the agent i when agent k is in its neighborhood and deemed an unreliable one. On the contrary, when the clipping step is taken, intuitively, we have to clip the agents that

1. Note that the delay varies in the asynchronous learning scheme, and there are two types of asynchronization, (i) fixed value of delays [18, 24] and (ii) time-varying delays [4, 11] along the learning process. We follow the first setting in this work to implement the experiments.

possess outdated information, resulting in the change of the mixing matrix $\underline{W}$. Essentially, we can manipulate the corresponding weight values $w_{ij}, j \in \mathcal{R}^c$ in $\underline{W}$ such that at this step, $w_{ij} = 0, j \in \mathcal{R}^c$. For the convenience of analysis, we introduce $\underline{\tilde{W}}$ to represent the mixing matrix, which follows the same properties in Assumption 1 (See Appendix for more detail).

Algorithm 1 PC-ASGD

```
Input: number of agents $N$, learning rate $\eta > 0$, agent interaction matrices $\underline{W}$, $\underline{\tilde{W}}$, number of
    epochs $T$, the tradeoff parameter $0 \leq \theta_t \leq 1, t \in \{0, 1, ..., T-1\}$
Output: the models' parameters in agents $x_T^i, i = 1, 2, ...N$
1: Initialize all the agents' parameters $x_0^i$, $i = 1, 2, ...N$
2: Do broadcast to identify the clusters of reliable agents and the delay $\tau$
3: $t = 0$
4: while epoch $t < T$ do
5:   for each agent $i$ do
6:     Predicting Step: $x_{t+1,pre}^i = \sum_{j\in\mathcal{R}} w_{ij} x_t^j - \eta g_i(x_t^i) + \sum_{k\in\mathcal{R}^c} w_{ik}(x_t^k - \eta g_k^{dc,r}(x_{t-\tau}^k))$
7:     Clipping Step: $x_{t+1,cli}^i = \sum_{j\in Nb(i)} \tilde{w}_{ij} x_t^j - \eta g_i(x_t^i)$
8:     $x_{t+1}^i = \theta_t x_{t+1,pre}^i + (1-\theta_t) x_{t+1,cli}^i$
9:   $t = t + 1$
```

We next investigate the convergence for the non-convex objectives. For PC-ASGD, we show that it converges to a first-order stationary point in a sublinear rate.

Theorem 2 *Let Assumption 1 hold. Assume that the delay compensated gradients are uniformly bounded, i.e., there exists a a scalar $B > 0$ such that for all $T \geq 1$ $\|\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \leq B, \ \forall t \geq 0$ and $0 \leq r \leq \tau - 1$, and there exists M, $\mathbb{E}[\|\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2] \leq M$. For the iterations generated by PC-ASGD, there exists $0 < \eta < \frac{1}{\gamma_m}$, $\gamma_m := \max\{\gamma_1, \gamma_2, ..., \gamma_N\}$, such that*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(\mathbf{x}_t)\|^2] \leq \frac{2(F(\mathbf{x}_1) - F^*)}{T\eta} + \frac{R}{\eta}, \tag{2}$$

where, $R = 2G\eta^2 C_1 + \frac{\tau^2\eta^2\gamma_m M}{2} + \frac{\eta\sigma^2}{2} + \eta\sigma\tau B + 2\eta^2\gamma_m(\tau B + G)C_1, C_1 = \frac{G+(\tau-1)B\theta_m}{1-\delta_2}$.

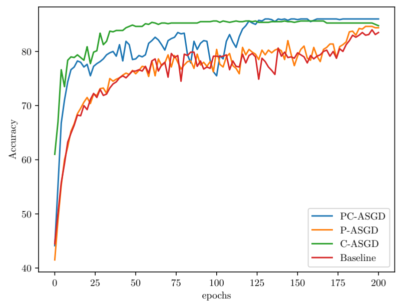
Remark 3 *Theorem 2 states that with a properly chosen constant step size, PC-ASGD is able to converge the iterates $\{\mathbf{x}_T\}$ to the noisy neighborhood of a stationary point $\mathbf{x}^*$ in a rate of $O(\frac{1}{\sqrt{T}})$, whose radius is determined by $\frac{\sigma^2}{2} + \sigma\tau B$, if we define $\eta = \mathcal{O}(\frac{1}{\sqrt{T}})$. Additionally, based on $\frac{\sigma^2}{2} + \sigma\tau B$, we can know that the error bound is mainly caused by the variance of stochastic gradients and the time delay.*

4. Experiments

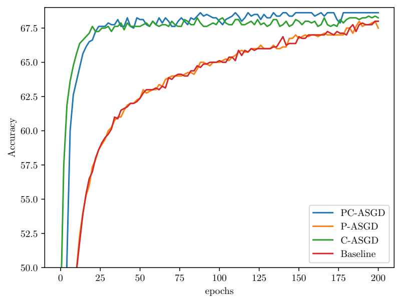
We have analyzed theoretically in detail how the proposed PC-ASGD converges with some mild assumptions. In practical implementation, we need to choose a suitable θ_t to enable synergy between clipping and predicting steps. In this context, we develop a heuristic practical variant with a criterion for determining the tradeoff parameter value. Intuitively, if the delay messages from the unreliable neighbors do not influence the training negatively, they should be included in the prediction. This can be determined by the comparison within the algorithm. The criterion is shown as follows by using the *cosine distance*:

$$x_i^{t+1} = \begin{cases} x_{t+1,pre}^i & \frac{\langle x_{t+1,pre}^i - x_t^i, g_i(x_t^i)\rangle}{\|x_{t+1,pre}^i - x_t^i\|} \geq \frac{\langle x_{t+1,cli}^i - x_t^i, g_i(x_t^i)\rangle}{\|x_{t+1,cli}^i - x_t^i\|} \\ x_{t+1,cli}^i & o.w. \end{cases} \tag{3}$$

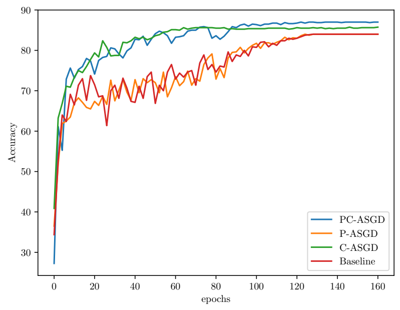
The prediction step is selected if it has the larger cosine distance, which implies that the update due to the predicting step yields the larger loss descent. Otherwise, the clipping step should be chosen by only trusting reliable neighbors. Please see Algorithm 2 in Appendix for detail.



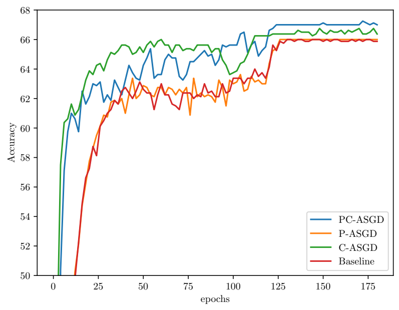
(a) DenseNet CIFAR-10



(b) DenseNet CIFAR-100



(c) PreResNet110 CIFAR-10



(d) PreResNet110 CIFAR-100

Figure 1: Testing accuracy on CIFAR-10 and CIFAR-100 with *distributed network 1*.

The testing accuracies on the CIFAR-10 and CIFAR-100 data sets with models in *distributed network 1* are shown in Fig. 1. It shows that the proposed PC-ASGD outperforms the other single variants and it presents an accuracy increment greater than 2.3% (nearly 4% for DenseNet with CIFAR-10) compared to the baseline algorithm (D-ASGD). For other variants P-ASGD or C-ASGD (variants from PC-ASGD), the testing accuracies are also higher than that of the baseline algorithm. Moreover, PC-ASGD shows faster convergence than P-ASGD as the updating rule overcomes the staleness, and achieves better accuracy than the C-ASGD as it includes the messages from the unreliable neighbors. This is consistent with the analysis in this work. We also show the detailed results of both *distributed network 1* (8 agents) and *distributed network 2* (20 agents) in Table 2. We then compare our proposed algorithm with other delay-tolerant algorithms, including the baseline algorithm D-ASGD (aka AD-PSGD), DC-s3gd [18], D-ASGD with IS [5], and Adaptive Braking [19]. The *distributed network 1* is applied for the comparisons. From the Table 3, the proposed PC-ASGD obtains the best results in the four cases.

Table 2: Performance evaluation of PC-ASGD on CIFAR-10 and CIFAR-100

8 agents							
Model & dataset	PC-ASGD		P-ASGD		C-ASGD		Baseline
	acc. (%)	o.p. (%)	acc. (%)	o.p. (%)	acc.(%)	o.p. (%)	acc. (%)
Pre110, CIFAR-10	$\mathbf{87.3 \pm 1.1}$	$\mathbf{3.3 \pm 1.1}$	84.9 ± 0.9	0.9 ± 0.9	86.0 ± 1.0	2.0 ± 1.0	84.0 ± 0.3
Pre110, CIFAR-100	$\mathbf{67.4 \pm 1.4}$	$\mathbf{3.1 \pm 1.9}$	64.8 ± 1.3	1.3 ± 1.5	66.4 ± 1.2	1.9 ± 1.6	64.5 ± 1.5
Des, CIFAR-10	$\mathbf{86.9 \pm 0.9}$	$\mathbf{3.6 \pm 1.8}$	84.4 ± 0.6	1.0 ± 1.5	85.9 ± 0.9	2.7 ± 1.7	83.3 ± 0.9
Des, CIFAR-100	$\mathbf{68.6 \pm 0.6}$	$\mathbf{2.3 \pm 1.7}$	66.8 ± 1.5	1.6 ± 1.6	66.8 ± 1.6	1.8 ± 1.6	66.1 ± 1.9

20 agents							
Model & dataset	PC-ASGD		P-ASGD		C-ASGD		Baseline
	acc. (%)	o.p. (%)	acc. (%)	o.p. (%)	acc.(%)	o.p. (%)	acc. (%)
Pre110, CIFAR-10	$\mathbf{84.7 \pm 0.9}$	$\mathbf{4.2 \pm 1.0}$	83.3 ± 0.9	2.7 ± 0.9	82.5 ± 1.0	1.9 ± 1.4	80.4 ± 0.7
Pre110, CIFAR-100	$\mathbf{62.4 \pm 0.8}$	$\mathbf{3.3 \pm 2.0}$	61.7 ± 1.0	2.0 ± 1.6	61.5 ± 1.0	2.5 ± 2.3	59.3 ± 1.7
Des, CIFAR-10	$\mathbf{82.9 \pm 0.9}$	$\mathbf{2.4 \pm 0.9}$	82.0 ± 0.7	1.4 ± 1.3	81.8 ± 0.6	1.8 ± 1.0	80.1 ± 0.9
Des, CIFAR-100	$\mathbf{64.5 \pm 0.7}$	$\mathbf{3.8 \pm 1.7}$	62.5 ± 1.3	2.9 ± 2.0	62.0 ± 1.5	1.3 ± 1.4	60.4 ± 1.7

acc.–accuracy, o.p.–outperformed comparing to baseline.

Table 3: Performance comparison for different delay tolerant algorithms

Model & dataset	Pre110,CIFAR-10	Pre110,CIFAR-100	Des,CIFAR-10	Des,CIFAR-100
PC-ASGD	$\mathbf{87.3 \pm 1.1}$	$\mathbf{67.4 \pm 1.4}$	$\mathbf{86.9 \pm 0.6}$	$\mathbf{68.6 \pm 0.6}$
D-ASGD [11]	84.0 ± 0.3	64.5 ± 1.5	83.3 ± 0.9	66.1 ± 1.9
DC-s3gd [18]	86.3 ± 0.8	63.5 ± 1.7	85.7 ± 0.8	66.2 ± 1.3
D-ASGD with IS [5]	85.0 ± 0.3	64.6 ± 1.2	84.6 ± 0.4	66.2 ± 0.8
Adaptive Braking [19]	86.8 ± 0.9	66.5 ± 1.2	85.3 ± 1.0	67.3 ± 1.1

5. Conclusion

This paper presents a novel learning algorithm for distributed deep learning with heterogeneous delay characteristics in agent-communication-network systems. We propose PC-ASGD algorithm consisting of a predicting step, a clipping step, and the corresponding update law for reducing the staleness and negative effects caused by the outdated weights. We present theoretical analysis for the convergence rate of the proposed algorithm with constant step size when the objective functions are weakly strongly-convex and nonconvex. The numerical studies show the effectiveness of our proposed algorithms in different distributed systems with delays, by comparing it to multiple baselines. In future work, the cases for distributed networks with diverse delays and dynamic topology will be further studied and tested.

References

[1] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. *arXiv: Optimization and Control*, 2011.

[2] Mahmoud S Assran and Michael G Rabbat. Asynchronous gradient push. *IEEE Transactions on Automatic Control*, 66(1):168–183, 2020.

[3] Xuanyu Cao, Tamer Başar, Suhas Diggavi, Yonina C Eldar, Khaled B Letaief, H Vincent Poor, and Junshan Zhang. Communication-efficient distributed learning: An overview. *IEEE journal on selected areas in communications*, 2023.

[4] Jeffrey Dean, Greg S Corrado, Rajat Monga, Kai Chen, and Andrew Y Ng. Large scale distributed deep networks. *Advances in neural information processing systems*, 2013.

[5] Yubo Du, Keyou You, and Yilin Mo. Asynchronous stochastic gradient descent over decentralized datasets. In *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, pages 216–221. IEEE, 2020.

[6] Hamid Reza Feyzmahdavian, Arda Aytekin, and Mikael Johansson. An asynchronous minibatch algorithm for regularized stochastic optimization. *conference on decision and control*, 61(12):1384–1389, 2015.

[7] Ning Gao, Le Liang, Donghong Cai, Xiao Li, and Shi Jin. Coverage control for uav swarm communication networks: A distributed learning approach. *IEEE Internet of Things Journal*, 9(20):19854–19867, 2022.

[8] Hubert Gijzen. Big data for a sustainable future. *Nature*, 502(7469):38–38, 2013.

[9] Zhanhong Jiang, Aditya Balu, Chinmay Hegde, and Soumik Sarkar. Collaborative deep learning in fixed topology networks. In *Advances in Neural Information Processing Systems*, pages 5904–5914, 2017.

[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[11] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. *arXiv: Optimization and Control*, 2017.

[12] Jing J Liang, A Kai Qin, Ponnuthurai N Suganthan, and S Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE transactions on evolutionary computation*, 10(3):281–295, 2006.

[13] Xiaolan Liu and Yuanwei Liu. Distributed learning for metaverse over wireless networks. *IEEE Communications Magazine*, 2023.

[14] Qinyi Luo, Jiaao He, Youwei Zhuo, and Xuehai Qian. Prague: High-performance heterogeneity-aware asynchronous decentralized training. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 401–416, 2020.

[15] Sudhanshu K Mishra. Some new test functions for global optimization and performance of repulsive particle swarm method. *Available at SSRN 926132*, 2006.

[16] Liangxin Qian, Ping Yang, Ming Xiao, Octavia A Dobre, Marco Di Renzo, Jun Li, Zhu Han, Qin Yi, and Jiarong Zhao. Distributed learning for wireless communications: Methods, applications and challenges. *IEEE Journal of Selected Topics in Signal Processing*, 16(3):326–342, 2022.

[17] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24:693–701, 2011.

[18] Alessandro Rigazzi. Dc-s3gd: Delay-compensated stale-synchronous sgd for large-scale decentralized neural network training. *arXiv: Learning*, 2019.

[19] Abhinav Venigalla, Atli Kosson, Vitaliy Chiley, and Urs Köster. Adaptive braking for mitigating gradient delay. *arXiv preprint arXiv:2007.01397*, 2020.

[20] Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Compact and computationally efficient representation of deep neural networks. *IEEE transactions on neural networks and learning systems*, 31(3):772–785, 2019.

[21] Xuyang Wu, Changxin Liu, Sindri Magnusson, and Mikael Johansson. Delay-agnostic asynchronous distributed optimization. *arXiv preprint arXiv:2303.18034*, 2023.

[22] Guojun Xiong, Gang Yan, Shiqiang Wang, and Jian Li. Straggler-resilient decentralized learning via adaptive asynchronous updates. *arXiv preprint arXiv:2306.06559*, 2023.

[23] Wei Yang. pytorch-classification. `https://github.com/bearpaw/pytorch-classification`, 2019. Accessed: 2019-01-24.

[24] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation. In *International Conference on Machine Learning*, pages 4120–4129. PMLR, 2017.

[25] Huiping Zhuang, Yi Wang, Qinglai Liu, and Zhiping Lin. Fully decoupled neural network learning using delayed gradients. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Appendix A. Additional Analysis

A.1. Derivation of delay compensated gradient $g_k^{dc,r}(x_{t-\tau}^k)$

The delay compensated gradient can be expressed as follows:

$$g_k^{dc,r}(x_{t-\tau}^k) = \sum_{r=0}^{\tau-1} g_k(x_{t-\tau}^k) + \lambda g_k(x_{t-\tau}^k) \odot g_k(x_{t-\tau}^k) \odot (x_{t-\tau+r}^i - x_{t-\tau}^i), \tag{4}$$

where λ is a positive constant in $(0,1]$ and the term $\lambda g_k(x_{t-\tau}^k) \odot g_k(x_{t-\tau}^k)$ is an estimate of the Hessian matrix, $\nabla g_k(x_{t-\tau}^k)$. We detail how to arrive at Eq. 4. Specifically, given the outdated weights of agent k, $x_{t-\tau}^k$, due to the delay equal to τ, by induction, we can obtain for agent k

$$\begin{aligned} x_{t-\tau+1}^k =& x_{t-\tau}^k - \eta g_k(x_{t-\tau}^k) \\ =& x_{t-\tau}^k - \eta \sum_{r=0}^{0}[g_k(x_{t-\tau}^k) + \lambda g_k(x_{t-\tau}^k) \odot g_k(x_{t-\tau}^k) \odot (x_{t-\tau+r}^i - x_{t-\tau}^i)] \end{aligned} \tag{5}$$

$$\begin{aligned} x_{t-\tau+2}^k &= x_{t-\tau+1}^k - \eta g_k(x_{t-\tau+1}^k) = x_{t-\tau}^k - \eta g_k(x_{t-\tau}^k) - \eta g_k(x_{t-\tau+1}^k) \\ &\approx x_{t-\tau}^k - \eta \sum_{r=0}^{1}[g_k(x_{t-\tau}^k) + \lambda g_k(x_{t-\tau}^k) \odot g_k(x_{t-\tau}^k) \odot (x_{t-\tau+r}^i - x_{t-\tau}^i)] \end{aligned} \tag{6}$$

$$\cdots$$

$$x_t^k \approx x_{t-\tau}^k - \eta \sum_{r=0}^{\tau-1}[g_k(x_{t-\tau}^k) + \lambda g_k(x_{t-\tau}^k) \odot g_k(x_{t-\tau}^k) \odot (x_{t-\tau+r}^i - x_{t-\tau}^i)] \tag{7}$$

As we mentioned in the main contents, the term $(x_{t-\tau+r}^i - x_{t-\tau}^i)$ is from agent i due to the outdated information of agent k, which intuitively illustrates that the compensation is driven by the agent i when agent k is in its neighborhood and deemed an *unreliable one*.

We now present convergence results for the PC-ASGD. We show the consensus estimate and the optimality for nonconvex smooth objectives. The consensus among agents (aka, disagreement estimate) can be thought of as the norms $\|x_t^i - x_t^j\|$, the differences between the iterates x_t^i and x_t^j. Alternatively, the consensus can be measured with respect to a reference sequence, i.e., $y_t = \frac{1}{N}\sum_{i=1}^N x_t^i$. In particular, we discuss $\|x_t^i - y_t\|$ for any time t as the metrics with respect to the delay τ.

Lemma 4 ***(Consensus)*** *Let Assumption 1 hold. Suppose that the delay compensated gradients are uniformly bounded, i.e., there exists a scalar $B > 0$, such that*

$$\|\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \leq B, \;\; \forall t \geq 0 \;\; and \;\; 0 \leq r \leq \tau - 1,$$

Then for all $i \in V$ and $t \geq 0$, $\exists \eta > 0$, we have

$$\mathbb{E}[\|x_t^i - y_t\|] \leq \eta \frac{G + (\tau-1)B\theta_m}{1-\delta_2}, \tag{8}$$

where $\theta_m = max\{\theta_{s+1}\}_{s=t}^{t+\tau-1}$, $\delta_2 = max\{\theta_s e_2 + (1-\theta_s)\tilde{e}_2\}_{s=0}^{t+\tau-1} < 1$, where $e_2 := e_2(W) < 1$ and $\tilde{e}_2 := e_2(\tilde{W}) < 1$, where $e_2(\cdot)$ and $\tilde{e}_2(\cdot)$ are the second-largest eigenvalues of W and $\tilde{W}$.

Proof Since

$$\begin{aligned}\|x^i_{t+\tau} - y_{t+\tau}\| &\leq \|\mathbf{x}_{t+\tau} - y_{t+\tau}\mathbf{1}\| \\ &= \|\mathbf{x}_{t+\tau} - \frac{1}{N}\mathbf{1}^T\mathbf{x}_{t+\tau}\mathbf{1}\| \\ &= \|\mathbf{x}_{t+\tau} - \frac{1}{N}\mathbf{1}\mathbf{1}^T\mathbf{x}_{t+\tau}\| \\ &= \|(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)\mathbf{x}_{t+\tau}\|,\end{aligned} \tag{9}$$

where $\mathbf{1}$ is the column vector with entries all being 1. According to Assumption 2, we have $\frac{1}{N}\mathbf{1}\mathbf{1}^T\mathcal{W} = \frac{1}{N}\mathbf{1}\mathbf{1}^T$. Hence, by induction, setting $\mathbf{x}_0 = 0$, and Lemma 1, the following relationship can be obtained

$$\begin{aligned}&\|\mathbf{x}_{t+\tau} - y_{t+\tau}\mathbf{1}\| = \eta\|\sum_{s=0}^{t+\tau-1}(\prod_{v=s+1}^{t+\tau-1}\mathcal{W}_{t+\tau+s-v} - \frac{1}{N}\mathbf{1}\mathbf{1}^T)\mathbf{g}(\mathbf{x}_s) \\ &+ \sum_{s=t}^{t+\tau-1}(\prod_{v=s+1}^{t+\tau-1}\mathcal{W}_{t+\tau+s-v} - \frac{1}{N}\mathbf{1}\mathbf{1}^T)\theta_{s+1}\sum_{r=0}^{\tau-2}W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \\ &\leq \eta\sum_{s=0}^{t+\tau-1}\|\prod_{v=s+1}^{t+\tau-1}\mathcal{W}_{t+\tau+s-v} - \frac{1}{N}\mathbf{1}\mathbf{1}^T\|\|\mathbf{g}(\mathbf{x}_s)\| + \eta\sum_{s=t}^{t+\tau-1}\|\prod_{v=s+1}^{t+\tau-1}\mathcal{W}_{t+\tau+s-v} - \frac{1}{N}\mathbf{1}\mathbf{1}^T\| \\ &\|\theta_{s+1}\sum_{r=0}^{\tau-2}W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \leq \eta G\sum_{s=0}^{t+\tau-1}\delta_2^{t+\tau-1-s} + \eta\sum_{s=t}^{t+\tau-1}\delta_2^{t+\tau-1-s}\theta_{s+1}(\tau-1)B \\ \leq&\eta G\frac{1}{1-\delta_2} + \eta(\tau-1)B\theta_m\frac{\delta_2^t - \delta_2^{t+\tau-1}}{1-\delta_2} \leq \eta\frac{G+(\tau-1)B\theta_m}{1-\delta_2}.\end{aligned} \tag{10}$$

The second inequality follows from the Triangle inequality and Cauthy-Schwartz inequality and the third inequality follows from Assumption 1 2) and that the matrix $\frac{1}{N}\mathbf{1}\mathbf{1}^T$ is the projection of $\mathcal{W}$ onto the eigenspace associated with the eigenvalue equal to 1. The last inequality follows from the property of geometric sequence. The proof is completed by replacing $t+\tau$ with t on the left hand side. ■

Lemma 4 states the consensus bound among agents, which is proportional to the step size η and inversely proportional to the gap between the largest and the second-largest magnitude eigenvalues of the equivalent graph $\mathcal{W}$.

Remark 5 *One implication that can be made from Lemma 4 is when $\tau = 1$, the consensus bound becomes the smallest, which can be obtained as $\frac{\eta G}{1-\delta_2}$. This bound is the same as obtained already by most decentralized learning (or optimization) algorithms. This accordingly implies that the delay compensated gradient or predicted gradient does not necessarily require many time steps. Otherwise, more compounding error could be included. Alternatively, $\theta_m = 0$ can also result in such a bound, suggesting that the clipping step dominates in the update. On the other hand, once $\tau \gg 1$ and $\theta_m \neq 0$, the consensus bound becomes worse, which will be validated by the empirical results. Additionally, if the network is sparse, which suggests $e_2 \to 1$ and $\tilde{e}_2 \to 1$, the consensus among agents may not be achieved well and correspondingly the optimality would be negatively affected, which has been justified in existing works [9].*

Appendix B. Proof of Theorem 2

Theorem 2: Let Assumptions 1,2 and 3 hold. Assume that the delay compensated gradients are uniformly bounded, i.e., there exits a a scalar $B > 0$ such that

$$\|\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \leq B, \ \ \forall t \geq 0 \ and \ 0 \leq r \leq \tau - 1, \tag{11}$$

and that

$$\mathbb{E}[\|\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2] \leq M. \tag{12}$$

Then for the iterates generated by PC-ASGD, there exists $0 < \eta < \frac{1}{\gamma_m}$, such that for all $T \geq 1$,

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(\mathbf{x}_t)\|^2] \leq \frac{2(F(\mathbf{x}_1) - F^*)}{T\eta} + \frac{R}{\eta}, \tag{13}$$

where

$$R = 2G\eta^2 C_1 + \frac{\tau\eta^2\gamma_m M}{2} + \frac{\eta\sigma^2}{2} + \eta\sigma\tau B + 2\eta^2\gamma_m(\tau B + G)C_1.$$

Proof According to the smoothness condition of $F(\mathbf{x})$, we have

$$\begin{aligned}
&F(\mathbf{x}_{t+\tau+1}) - F(\mathbf{v}_{t+\tau}) \\
&\leq \langle \nabla F(\mathbf{v}_{t+\tau}), \mathbf{x}_{t+\tau+1} - \mathbf{v}_{t+\tau}\rangle + \frac{\gamma_m}{2} + \|\mathbf{x}_{t+\tau+1} - \mathbf{v}_{t+\tau}\|^2 \\
&= \langle \nabla F(\mathbf{v}_{t+\tau}), -\eta(\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau}))\rangle + \frac{\eta^2\gamma_m}{2}\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r} + \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 \\
&= \langle \nabla F(\mathbf{v}_{t+\tau}) - \nabla F(\mathbf{x}_{t+\tau}) + \nabla F(\mathbf{x}_{t+\tau}), \eta(\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau}))\rangle \\
&+ \frac{\eta^2\gamma_m}{2}\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r} + \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 = -\eta\langle \nabla F(\mathbf{x}_{t+\tau}), \sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\rangle \\
&+ \eta\langle(\nabla F(\mathbf{v}_{t+\tau}) - \nabla F(\mathbf{x}_{t+\tau}), \sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau}))\rangle \\
&+ \frac{\eta^2\gamma_m}{2}\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r} + \mathbf{g}(\mathbf{x}_{t+\tau})\|^2
\end{aligned} \tag{14}$$

Hence, we have

$$
\begin{aligned}
&F(\mathbf{x}_{t+\tau+1}) - F(\mathbf{v}_{t+\tau}) \\
&= -\frac{\eta}{2}[\|\nabla F(\mathbf{x}_{t+\tau})\|^2 + \|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 \\
&- \|\nabla F(\mathbf{x}_{t+\tau}) - (\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau}))\|^2] + \eta\langle\nabla F(\mathbf{x}_{t+\tau}) - \nabla F(\mathbf{v}_{t+\tau}), \sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\rangle \\
&+ \frac{\eta^2\gamma_m}{2}\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r} + \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 = -\frac{\eta}{2}\|\nabla F(\mathbf{x}_{t+\tau})\|^2 - \frac{\eta}{2}\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 \\
&+ \frac{\eta}{2}(\|\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 + \|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2 - 2\langle\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau}), \sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\rangle) \\
&+ \eta\langle\nabla F(\mathbf{x}_{t+\tau}) - \nabla F(\mathbf{v}_{t+\tau}), \sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\rangle + \frac{\eta^2\gamma_m}{2}\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r} + \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 \\
&= -\frac{\eta}{2}\|\nabla F(\mathbf{x}_{t+\tau})\|^2 - (\frac{\eta}{2} - \frac{\eta^2\gamma_m}{2})\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 \\
&+ \frac{\eta}{2}\|\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 + \frac{\eta}{2}\|\sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2 - \eta\langle\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau}), \sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\rangle \\
&+ \eta\langle\nabla F(\mathbf{x}_{t+\tau}) - \nabla F(\mathbf{v}_{t+\tau}), \sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\rangle = -\frac{\eta}{2}\|\nabla F(\mathbf{x}_{t+\tau})\|^2 + (\frac{\eta^2\gamma_m}{2} - \frac{\eta}{2}) \\
&\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2 + (\frac{\eta^2\gamma_m}{2} - \frac{\eta}{2})\|\mathbf{g}(\mathbf{x}_{t+\tau})\|^2 \\
&+ (\frac{\eta^2\gamma_m}{2} - \frac{\eta}{2})\langle\mathbf{g}(\mathbf{x}_{t+\tau}), \sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\rangle \\
&+ \frac{\eta}{2}\|\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 + \frac{\eta}{2}\|\sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2 - \eta\langle\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau}), \\
&\sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\rangle + \eta\langle\nabla F(\mathbf{x}_{t+\tau}) - \nabla F(\mathbf{v}_{t+\tau}), \sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\rangle
\end{aligned}
\tag{15}
$$

We then have

$$\begin{aligned}
&F(\mathbf{x}_{t+\tau+1}) - F(\mathbf{v}_{t+\tau}) \\
&\leq -\frac{\eta}{2}\|\nabla F(\mathbf{x}_{t+\tau})\|^2 + (\frac{\eta^2\gamma_m}{2} - \frac{\eta}{2})\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2 \\
&+ (\frac{\eta^2\gamma_m}{2} - \frac{\eta}{2})\|\mathbf{g}(\mathbf{x}_{t+\tau})\|^2 + (\frac{\eta^2\gamma_m}{2} - \frac{\eta}{2})\|\mathbf{g}(\mathbf{x}_{t+\tau})\|\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \\
&+ \frac{\eta}{2}\|\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau})\|^2 + \frac{\eta}{2}\|\sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2 + \eta\|\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau})\|\|\sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\| \\
&+ \eta\|\nabla F(\mathbf{x}_{t+\tau}) - \nabla F(\mathbf{v}_{t+\tau})\|\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\|.
\end{aligned} \tag{16}$$

The last inequality follows from Cauthy-Schwarz inequality.

The left hand side of the above inequality can be rewritten associated

$$F(\mathbf{x}_{t+\tau+1}) - F(\mathbf{x}_{t+\tau}) + F(\mathbf{x}_{t+\tau}) - F(\mathbf{v}_{t+\tau})$$

Taking expectation for both sides, with the last inequality, we have

$$\begin{aligned}
&\mathbb{E}[F(\mathbf{x}_{t+\tau+1}) - F(\mathbf{x}_{t+\tau})] \leq \mathbb{E}[F(\mathbf{v}_{t+\tau}) - F(\mathbf{x}_{t+\tau})] - \frac{\eta}{2}\mathbb{E}[\|\nabla F(\mathbf{x}_{t+\tau})\|^2] \\
&+ \frac{\eta^2\gamma_m - \eta}{2}\mathbb{E}[\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2] + \frac{\eta^2\gamma_m - \eta}{2}\mathbb{E}[\|\mathbf{g}(\mathbf{x}_{t+\tau})\|^2] \\
&+ \frac{\eta^2\gamma_m - \eta}{2}\mathbb{E}[\|\mathbf{g}(\mathbf{x}_{t+\tau})\|\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|] + \frac{\eta}{2}\mathbb{E}[\|\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau})\|^2] \\
&+ \frac{\eta}{2}\mathbb{E}[\|\sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2] + \eta\mathbb{E}[\|\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau})\|\|\sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|] \\
&+ \eta\mathbb{E}[\|\nabla F(\mathbf{x}_{t+\tau}) - \nabla F(\mathbf{v}_{t+\tau})\|\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\|] \\
\leq& G\mathbb{E}[\|\mathbf{v}_{t+\tau} - \mathbf{x}_{t+\tau}\|] - \frac{\eta}{2}\mathbb{E}[\|\nabla F(\mathbf{x}_{t+\tau})\|^2] + \frac{\eta^2\gamma_m - \eta}{2}\tau\sum_{r=0}^{\tau-1}\mathbb{E}[\|W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2] \\
&+ \frac{\eta^2\gamma_m - \eta}{2}\mathbb{E}[\|\mathbf{g}(\mathbf{x}_{t+\tau})\|^2] + \frac{\eta^2\gamma_m - \eta}{2}\mathbb{E}[\|\mathbf{g}(\mathbf{x}_{t+\tau})\|\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|] \\
&+ \frac{\eta}{2}\mathbb{E}[\|\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau})\|^2] + \frac{\eta}{2}\mathbb{E}[\|\sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|^2] \\
&+ \eta\mathbb{E}[\|\nabla F(\mathbf{x}_{t+\tau}) - \mathbf{g}(\mathbf{x}_{t+\tau})\|\|\sum_{r=1}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t)\|] + \eta\mathbb{E}[\|\nabla F(\mathbf{x}_{t+\tau}) - \nabla F(\mathbf{v}_{t+\tau})\| \\
&\|\sum_{r=0}^{\tau-1} W'\mathbf{g}^{dc,r}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_{t+\tau})\|] \leq -\frac{\eta}{2}\mathbb{E}[\|\nabla F(\mathbf{x}_{t+\tau})\|^2] \\
&+ \frac{\tau^2\eta^2\gamma_m M}{2} + \frac{\eta\sigma^2}{2} + \eta\sigma\tau B + 2\eta^2\gamma_m(\tau B + G + \frac{G}{\eta\gamma_m})\frac{G + (\tau-1)B\theta_m}{1-\delta_2}
\end{aligned}$$

The last inequality follows from the smoothness condition of $F(\mathbf{x})$ and the bounded gradient, respectively, as well as $\eta < \frac{1}{\gamma_m}$. Hence, by replacing $t+\tau$ with t, one can obtain

$$\mathbb{E}[F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t)] \leq -\frac{\eta}{2}\mathbb{E}[\|\nabla F(\mathbf{x}_t)\|^2] + R \tag{17}$$

where R indicates the constant term on the right hand side of the inequality. As we assume that $F(\mathbf{x})$ is bounded from below, applying the last inequality from 1 to T, one can get

$$\begin{aligned}
F^* - F(\mathbf{x}_1) &\leq \mathbb{E}[F(\mathbf{x}_{t+1})] - F(\mathbf{x}_1) \\
&\leq -\frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(\mathbf{x}_t)\|^2] + TR
\end{aligned} \tag{18}$$

which results in

$$\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(\mathbf{x}_t)\|^2] \leq \frac{2[(F(\mathbf{x}_1) - F^*) + TR]}{\eta} \tag{19}$$

Dividing both sides by T, the desirable results is obtained. ■

Appendix C. Settings

Our practical variant with this criterion in Eq. 3 still converges since we just set θ_t as 0 or 1 for each iteration and the previous analysis in our paper still holds. To facilitate the understanding of predicting and clipping steps, in the experiments, we also have two other variants P-ASGD and C-ASGD. While the former corresponds to an "optimistic" scenario to only rely on the predicting step, the latter presents a "pessimistic" scenario by dropping all outdated agents. Both of variants follow the same convergence rates induced by PC-ASGD. The specific algorithm is showed as Algorithm 2.

Algorithm 2 PC-ASGD-PV

```
Input: number of agents $N$, learning rate $\eta > 0$, agent interaction matrices $\underline{W}$, $\underline{\tilde{W}}$, number of
    epochs $T$
Output: the models' parameters in agents $x^i_T$,$i = 1, 2, ...N$
 1: Initialize all the agents' parameters $x^i_0$, $i = 1, 2, ...N$
 2: Do broadcast to identify the clusters of reliable agents and the delay $\tau$
 3: $t = 0$
 4: while epoch $t < T$ do
 5:   for each agent $i$ do
 6:     Predicting Step: $x^i_{t+1,pre} = \sum_{j\in\mathcal{R}} w_{ij} x^j_t - \eta g_i(x^i_t) + \sum_{k\in\mathcal{R}^c} w_{ik}(x^k_t - \eta g^{dc}_k(x^k_{t-\tau}))$
 7:     Clipping Step: $x^i_{t+1,cli} = \sum_{j\in\mathcal{R}} \tilde{w}_{ij} x^j_t - \eta g_i(x^i_t)$
 8:     $\Delta_{pre} = x^i_{t+1,pre} - x^i_t$; $\Delta_{cli} = x^i_{t+1,cli} - x^i_t$
 9:     if $\frac{\langle \Delta_{pre}, g_i(x^i_t)\rangle}{\|\Delta_{pre}\|} \geq \frac{\langle \Delta_{cli}, g_i(x^i_t)\rangle}{\|\Delta_{cli}\|}$ then
10:       $x^i_{t+1} = x^i_{t+1,pre}$
11:     else
12:       $x^i_{t+1} = x^i_{t+1,cli}$
13:   $t = t + 1$
```

C.1. Distributed Network and Learning Setting

Models and Data sets. Decentralized asynchronous SGD (D-ASGD) is adopted as the baseline algorithm. Two deep learning structures, PreResNet110 and DenseNet (noted as *model 1* and *model 2*), are employed. The detailed model structures are illustrated in the Appendix. CIFAR-10 and CIFAR-100 are used in the experiments following the settings in [10]. The training data is randomly assigned to each agent, and the parameters of the deep learning structure are maintained within each agent and communicated with the predefined delays. The testing set is utilized for each agent to verify the performance, where our metric is the average accuracy among the agents. 6 runs are carried out for each case and the mean and variance are obtained and listed in Table 3.

As for the practical implementation, an structure that is much closer to the real distributed system is used. Each agent is allocated to an independent GPU, and a communication layer is set up for the parameter transferring, which is convenient to the following protocol design. Such settings provide us availability for quick implementation of the algorithm in real distributed networks.

Delay setting. The delay is set as τ as discussed before, which means the parameters received from the agents outside of the reliable cluster are the ones that were obtained τ iterations before. For *model 1* and *model 2*, τ is both fixed at 20 to test the performances of different algorithms including our different variants (P-ASGD, C-ASGD, and PC-ASGD) and baseline algorithms.

Distributed network setting. A distributed network (noted as *distributed network 1*) with 8 agents (nodes) in a fully connected graph is first applied with *model 1* and *model 2*, and 2 clusters of reliable agents are defined within the graph consisting of 3 agents and 5 agents, respectively. Then another distributed network (20-agent) is used, noted as *distributed network 2* involving 4 clusters with 3 clusters consisting of 6 agents while the rest has 2 agents.

C.2. Detailed Settings of Deep Learning Models

For the PreResNet110 (*model 1*) and DenseNet (*model 2*), the batch size is selected as 128. After hyperparameter searching in $(0.1, 0.01, 0.001)$, the learning rate is set as 0.01 for the first 160 epochs and changed as 0.001. The decay are applied in epochs $(80, 120, 160, 200)$. The approximation coefficient λ is set as 1. $\lambda = 0.001$ is first tried as suggested by DC-ASGD [24] and the results show that the predicting step doesn't affect the training process. By considering the upper bound of 1, a set of values $(0.001, 0.1, 1)$ are tried and $\lambda = 1$ is applied according to the performance.

Appendix D. Additional Results

D.1. Impacts of Different Delay Settings

To further show our algorithm's effectiveness, we also implement experiments with different delays. As discussed above, a more severe delay could cause significant drop on the accuracy. More numerical studies with different steps of delay are presented here. The delays are set as $5, 20, 60$ with our PreResNet110 (*model 1*) of 8 agents (synchronous network without delay is also tested). We use CIFAR-10 in the studies and the topology is *distributed network 1*. The results are shown in Fig. 2.

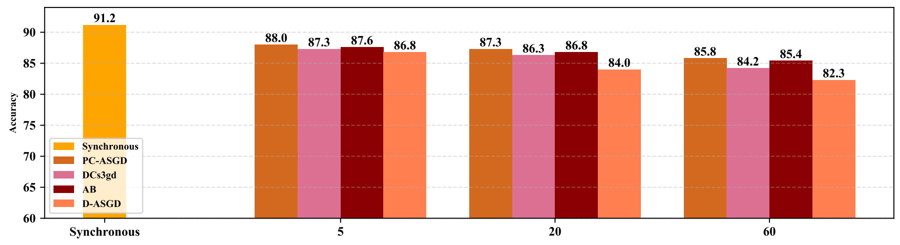


Figure 2: Performance evaluation for different steps of delay.

We can find out as the delay increases, the accuracy decreases. For the synchronous setting, the testing accuracy is close to that in the centralized scenario [23] but with higher batch size. When the delay is 60, the accuracy for the D-ASGD reduces significantly, and this validates that the large delay significantly influences the performance and causes difficulties in the training process. However, the delays are practical in the real implementations such as industrial IoT platforms.

For our proposed PC-ASGD, it outperforms other algorithms in all cases with different delays. Moreover, the accuracy drop is relatively smaller in cases with larger delays, which suggests that PC-ASGD is more robust to different communication delays.

D.2. Numerical Studies on θ Assignments

We also conduct empirical studies about the different choices for θ. As we mentioned above, a practical variant is applied for θ, where we intend to form a strategy to determine if the received information (parameters of the deep learning models) is outdated or not. Here, different assignment rules for θ are tested and compared. *Model 1* is applied, by using CIFAR-10 and the 8 agents system with 3 and 5 agents (*distributed network 1*).

First, θ is fixed as $0.3, 0.5, 0.7$ (denoted as *f1, f2, f3*), respectively. Then we determine the θ as $0, 1$ randomly with fixed probability in each round with $0.3, 0.5, 0.7$ (denoted as *p1, p2, p3*). We also try the fully uniformly random assigned θ in each round (denoted as *r1*). The results are listed

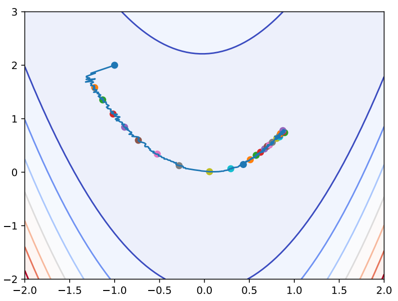
Table 4: Mean Performance for Different θ assignment for Pre110, CIFAR-10

Method\Parameters	*f1/p1*	*f2/p2*	*f3/p3*
θ Fixed	86.3	85.0	84.5
θ Bool randomly	85.6	85.0	84.1
θ randomly (r1)		85.2	
PC-ASGD-PV		**87.3**	
D-ASGD(Baseline)		84.0	

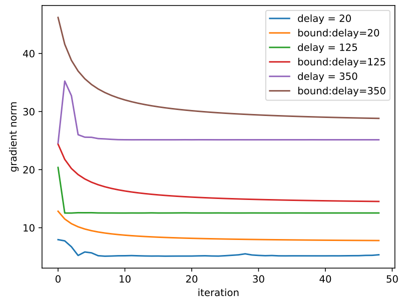
in Table 4. Note also that in the first three scenarios the criterion is not applied. The PC-ASGD-PV obtains the best performance which implies that the trade-off between the predicting step and the clipping step in the Algorithm 2 is proper and plays an important role in the convergence process. With the fixed θ (first row 'θ fixed'), the experimental results show that the optimal ratio between the predicting step and clipping step is 0.3 in this case. And this suggests that more clipping steps are better. For the *p1, p2, p3* cases (second row θ Bool randomly i.e. either 0 or 1), the experimental results show that the optimal probability between the predicting step and clipping step is 0.3. This is consistent with the fixed θ case. Compared with the fix θ setting, picking $0, 1$ for the θ in a predefined probability performs worse. The randomness still help the convergence process but is not as good as the fix θ setting. For the random θ, the randomness helps the convergence process. However, there exists a optimal θ for every case and the randomness is not able to get the best performance. The baseline D-ASGD gets the worst performance, which shows the predicting and clipping steps are helpful for the scenarios with delays in the distributed network. This also provides us the necessity of the additional time cost for the predicting and clipping steps. Note also that optimizing the selection of θ is beneficial and we can set θ as binary or non-binary (continuous). The binary setting with the strategy in Algorithm 2 is straightforward and performs well in this work.

D.3. Validation for Theoretical Analysis

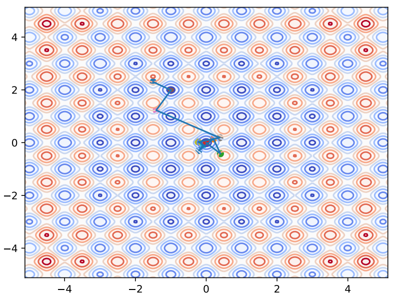
Finally, we present two examples to verify our constructed theoretical analysis. We establish a network involving three agents. We also set two reliable clusters with 1 and 2 agents, respectively. We leverage two nonconvex functions, i.e., Rastrigin and Rosenbrock [12] to test the performance of our proposed framework. Though these two functions are simple nonconvex problems, they have been used widely to test the performance for many numerical optimizers [15]. We randomly sample batch during local training in each agent. We set a fixed step size according our Theorem 2 as 0.008. The number of iterations is set 500 for each case.



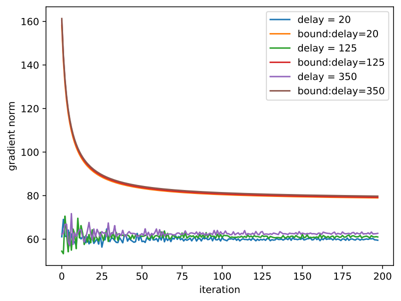
(*a*) Convergence trajectories for Rosenbrock function



(*b*) Average gradients bound verification in Rosenbrock function



(*c*) Convergence trajectories for Rastrigin function



(*d*) Average gradients bound verification in Rastrigin function

Figure 3: The results of simple functions.

From Fig. 3, we can view the convergence of our proposed PC-ASGD algorithms. For the bound verification, we take different values of the delay to observe the performances of our theoretical framework. Here, we first find that when delay is large, the squared norm of the gradient is large, which is consistent with our theoretical analysis. In Rosenbrock function case, our established theory could describe the tendency of the average gradients square norm and the results are nearly tight asymptotically. But in Rastrigin function cases, we observe that the differences between different delay are not large such that the bound is not so tight. However, when calculating bounds, we find that the bounds for different delays differ mildly, which is consistent along all the empirical results. It also shows the effectiveness of our proposed theoretical analysis.