OPT2024: 16th Annual Workshop on Optimization for Machine Learning

MindFlayer: Efficient Asynchronous Parallel SGD in the Presence of Heterogeneous and Random Worker Compute Times

Artavazd Maranjyan HTTPS://ARTOMARANJYAN.GITHUB.IO/
Omar Shaikh Omar OMAR.SHAIKHOMAR@KAUST.EDU.SA
Peter Richtárik
Center of Excellence for Generative AI, King Abdullah University of Science and Technology (KAUST) & SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence (SDAIA-KAUST AI) Thuwal, Kingdom of Saudi Arabia

Abstract

We study the problem of minimizing the expectation of smooth nonconvex functions with the help of several parallel workers whose role is to compute stochastic gradients. In particular, we focus on the challenging situation where the workers' compute times are arbitrarily heterogeneous and random. In the simpler regime characterized by arbitrarily heterogeneous but deterministic compute times, Tyurin and Richtárik [24] recently proposed the first optimal asynchronous SGD method, called Rennala SGD, in terms of a novel complexity notion called time complexity. The starting point of our work is the observation that Rennala SGD can have arbitrarily bad performance in the presence of random compute times – a setting it was not designed to handle. To advance our understanding of stochastic optimization in this challenging regime, we propose a new asynchronous SGD method, for which we coin the name MindFlayer SGD. Our theory and empirical results demonstrate the superiority of MindFlayer SGD over existing baselines, including Rennala SGD, in cases when the noise is heavy tailed.

1. Introduction

We address the nonconvex optimization problem:

$$\min_{x\in\mathbb{R}^d} \left\{ f(x) := \mathbb{E}_{\xi\sim\mathcal{D}}\left[f(x;\xi)\right] \right\}, \tag{1}$$

where $f : \mathbb{R}^d \times \mathbb{S} \to \mathbb{R}$, and ξ is a random variable with some distribution $\mathcal{D}$ on $\mathbb{S}$. In the context of machine learning, $\mathbb{S}$ could represent the space of all possible data, $\mathcal{D}$ denotes the distribution of the training dataset, and $f(\cdot,\xi)$ denotes the loss of a data sample ξ.

The function f is assumed to be differentiable, and its gradient is L–Lipschitz continuous (see Assumptions 1–2). We assume that we have n workers available to work in parallel, each able to compute independent, unbiased stochastic gradients of f, whose variance is bounded by σ^2 (see Assumption 3). In this paper, we are interested in investigating the time complexity of methods working in this natural setup.

1.1. Parallel Methods

In distributed systems with n clients computing stochastic gradients in parallel, the classic approach is Minibatch SGD [5, 10, 11]:

1. Receive stochastic gradients $\nabla f(x^k; \xi_i^k)$ from all workers $i \in [n]$,
2. Update the model: $x^{k+1} = x^k - \gamma \frac{1}{n} \sum_{i=1}^n \nabla f(x^k; \xi_i^k)$,

where $\gamma > 0$ is the stepsize, and ξ_i^k are i.i.d. samples. However, Minibatch SGD is inefficient in systems where workers have different compute times, as the overall step time is determined by the slowest worker.

Asynchronous SGD. To better use all computational resources, ASGD algorithms [1, 4, 7, 16, 20, 22, 23] update the model as soon as any worker completes its computation:

1. Receive $\nabla f(x^{k-\delta_k}; \xi^{k-\delta_k})$ from any worker,
2. Update: $x^{k+1} = x^k - \gamma \nabla f(x^{k-\delta_k}; \xi^{k-\delta_k})$,
3. Send x^{k+1} back to the worker for the next gradient.

While ASGD can reduce idle time, it can suffer from delays due to outdated gradients, negatively impacting convergence [24].

Rennala SGD. Tyurin and Richtárik [24] introduced Rennala SGD, which strikes a balance between efficiency and gradient staleness. It collects B gradients from workers at point x^k, where faster workers contribute more gradients:

1. Wait for B gradients at point x^k,
2. Update: $x^{k+1} = x^k - \gamma \frac{1}{B} \sum_{j=1}^B \nabla f(x^k; \xi_j^k)$.

This method is mini-max optimal for fixed compute times, but in practice, random compute times are more realistic.

While it may seem that the story is over, we want to question the fixed time assumption, arguing that a random time model is more realistic. The claim of optimality does not hold because of this randomness, suggesting that the algorithms need to be reevaluated and redesigned. We believe that a redesign is necessary to better fit this more realistic approach.

2. Problem Setup and Contributions

The deterministic compute time setup used by Tyurin and Richtárik [24] for Rennala SGD doesn't reflect the complexities of real-world distributed learning, where compute times are uncertain due to hardware failures, job preemptions, GPU delays, and network inconsistencies [3, 6]. This issue is even more severe in federated learning, where client unreliability leads to unpredictable compute times or incomplete tasks [14].

To tackle these challenges, we introduce a setup that accounts for random compute times. We define the stochastic gradient computation time for worker i as $\tau_i + \eta_i$, where $\tau_i > 0$ is the minimum compute time, and η_i is a non-negative random variable from distribution $\mathcal{J}_i$, representing the uncertainties.

In this more realistic setting, existing methods like Rennala SGD and ASGD can perform poorly or even fail to converge. We can illustrate this with a simple example:

Consider a scenario where each time we request a device to compute a stochastic gradient, one of two outcomes occurs. Either the device completes the computation exactly after the minimum

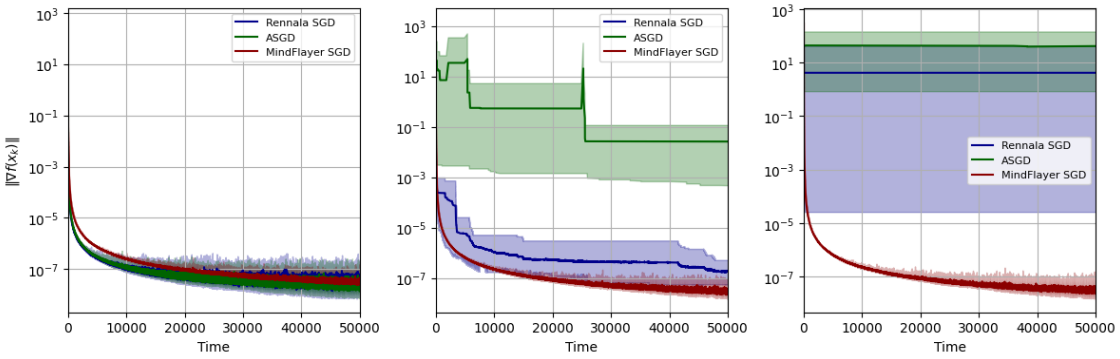

Figure 1: We ran an empirical experiment[2] where we employ the same $\mathcal{J}_i = \text{Lognormal}(0, s)$ distribution for all clients $i \in [n]$, with varying standard deviations s. Specifically, we set $s = 1$ for the left, $s = 10$ for the middle, and $s = 100$ for the right. Additionally, we set $\tau_i = \sqrt{i+1}$. As we observe, with an increase in the variance of the distribution, MindFlayer SGD demonstrates the ability to significantly outperform Rennala SGD and ASGD.

time τ without any delays, or something goes wrong and the computation is never completed. This situation can be modeled using a random time η as follows:

$$\eta = \begin{cases} 0, & \text{with probability } 1-q, \\ \infty, & \text{with probability } q, \end{cases} \tag{2}$$

where $0 < q < 1$. In this scenario, any method that waits for a certain number of batches on each iteration to perform a step runs the risk of never receiving the required batch and getting stuck. This includes methods like Rennala SGD or ASGD.

To overcome these limitations, we introduce, MindFlayer SGD [1] in Algorithm 1. Unlike Rennala SGD or ASGD, which wait for a fixed number of gradients (one in the case of ASGD), MindFlayer SGD allocates a specific time for computing a single stochastic gradient. If a client fails to complete its computation within the allotted time, we discard the incomplete computation and start anew.

We show that our method is a generalization of Rennala SGD, meaning that it is optimal in the deterministic compute time setup. Moreover, it can be arbitrarily faster than Rennala SGD or ASGD depending on the randomness in compute times. Particularly, we show that if the distribution is positively-skewed, then our method is faster, as shown in Figure 1 where $\mathcal{J}_i = \text{Lognormal}(0, s)$. As s gets bigger, the distribution's skewness coefficient gets bigger and the performance of Rennala SGD or ASGD gets worse. Meanwhile, our method MindFlayer SGD is robust to the change of the variance, as demonstrated by Theorem 5.

To the best of our knowledge, MindFlayer SGD is the first algorithm designed to work in the presence of heterogeneous and random worker compute times. In Section 3, we present the theoretical time complexity for this method and demonstrate its versatility in working with any choice of hyperparameters. Notably, it can operate without prior knowledge of the fixed times τ_i and the distributions $\mathcal{J}_i$.

1. We name our method MindFlayer SGD, drawing inspiration from The Mind Flayer from *Stranger Things*, due to its ability to precisely control its clients (Algorithm 2), analogous to the creature's control over its victims (The Flayed).
2. On a quadratic problem with $n = 5$ clients. We tuned stepsizes for all, and used theoretical trials B_i for MindFlayer SGD from Theorem 5 and tuned batch size for Rennala SGD, see Appendix I.

Furthermore, in Appendix D we expand our theory to develop Vecna SGD, tailored for the heterogeneous case, where workers have datasets that are coming from different distributions.

To illustrate our superior performance of MindFlayer SGD, we conduct an evaluation using various functions and distributions, as detailed in Appendix I. For distributions, we consider the Lognormal, Log-Cauchy, and the Infinite-Bernoulli (defined by Equation (2)) distributions. As for the functions, we consider a quadratic loss and a neural network on the MNIST dataset [18]. This diverse testing setup allows us to demonstrate MindFlayer SGD's robustness and effectiveness across a variety of challenging scenarios.

3. MindFlayer SGD

Here, we propose our MindFlayer SGD algorithm for multiple device case ($n > 1$). For the heterogeneous case, please refer to Appendix D.

Algorithm 1 MindFlayer SGD

Input: starting point $x^0 \in \mathbb{R}^d$, stepsize $\gamma > 0$, allotted times $t_1, \ldots, t_n \geq 0$, number of trials per client $B_1, \ldots, B_n \geq 0$
for $k = 1, 2, \ldots, K$ **do**
  Put $g^k = 0$
  Send x^k to all clients
  Run Method 2 in all clients $i = 1, 2, \ldots, n$
  while $\exists$ client that has trials to perform **do**
    Wait for the fastest client
    Receive gradient g
    $g^k = g^k + g$
  end while
  $g^k = \frac{g^k}{B}$, $\diamond$ $B = \sum_{i=1}^n p_i B_i$ and $p_i = F_i(t_i) = P(\eta_i \leq t_i)$.
  $x^{k+1} = x^k - \gamma g^k$
end for

Algorithm 2 Client i-s k-th step

Receive x^k from the server
for $j = 1, 2, \ldots, B_i$ **do**
  Sample $\eta_i^j \sim \mathcal{J}_i$
  if $\tau_i + \eta_i^j \leq t_i$ **then**
    $g = \nabla f(x^k; \xi_i^j)$, $\xi_i^j \sim \mathcal{D}$
    Send g to the server
  end if
end for

The MindFlayer SGD algorithm begins with an initialization at a starting point x^0 in $\mathbb{R}^d$, with a specified stepsize $\gamma > 0$, time allowances $t_i > 0$, and trial counts $B_i \geq 0$ for each client. In each iteration k, the server distributes the current point x^k to all clients. Each client i then executes a subroutine (Method 2) to attempt to compute B_i stochastic gradients. During each attempt, client i starts computing a stochastic gradient; if the computation exceeds the allotted time t_i, they discard the current gradient and begin another computation. Consequently, the actual number of stochastic gradients received from each client i is a random variable, ranging from 0 to B_i. The expected number of gradients from client i is given by $p_i B_i$, thus, the overall expected total of stochastic gradients is $B = \sum_{i=1}^n p_i B_i$. The server aggregates these received stochastic gradients and normalizes the collective gradient by the expected batch size B. Finally, the point is updated to $x^{k+1} = x^k - \gamma g^k$ following each aggregation round.

To give motivation in the design of our algorithm we consider the one device case in Section A. To continue with the analysis of MindFlayer SGD, we first present the assumptions under which this method is studied.

2. On a quadratic problem with theoretical hyperparameters, see Appendix I.

3.1. Assumptions

We consider standard assumptions used in the nonconvex optimization [9].

Assumption 1 *Function f is differentiable, and its gradient is L–Lipschitz continuous, i.e., $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$, for all $x, y \in \mathbb{R}^d$.*

Assumption 2 *There exist $f^* \in \mathbb{R}$ such that $f(x) \geq f^*$ for all $x \in \mathbb{R}^d$.*

Assumption 3 *For all $x \in \mathbb{R}^d$, stochastic gradients $\nabla f(x;\xi)$ are unbiased and σ^2-variance-bounded, i.e., $\mathbb{E}_\xi [\nabla f(x;\xi)] = \nabla f(x)$ and $\mathbb{E}_\xi \left[\|\nabla f(x;\xi) - \nabla f(x)\|^2\right] \leq \sigma^2$, where $\sigma^2 \geq 0$.*

3.2. Convergence theory

The following theorem gives iterations guarantees for the convergence of MindFlayer SGD.

Even though MindFlayer SGD is similar to Rennala SGD the convergence analysis require additional considerations, since the batch size is a random variable here as apposed to the case of Rennala SGD.

Theorem 4 (Proof in Appendix G.1) *Assume that Assumptions 1, 2 and 3 hold. Let $B = \sum_{i=1}^n p_i B_i$ and $\gamma = \frac{1}{2L} \min\left\{1, \frac{\varepsilon B}{\sigma^2}\right\}$ in Method 1. Letting $\Delta := f(x^0) - f^*$, we have that after*

$$K \geq \max\left\{1, \frac{\sigma^2}{\varepsilon B}\right\} \frac{8\Delta L}{\varepsilon}$$

iterations, the method guarantees that $\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2\right] \leq \varepsilon$.

Note that in the deterministic case where $\eta_i = 0$ for all $i \in [n]$, we have $p_i = P(\eta_i \leq t_i) = 1$ for all $i \in [n]$. Therefore, we derive $K \geq \max\left\{1, \frac{\sigma^2}{\varepsilon B}\right\} \frac{8\Delta L}{\varepsilon}$, with $B = \sum_{i=1}^n B_i$, yielding the same result as in Rennala SGD.

We also achieve the same rate as $t_i \to \infty$ for all i, since in that scenario $p_i \to 1$. This is expected because we will observe a consistent number of stochastic gradients each time, though the timing may vary, as mentioned earlier.

However, if $t_i = 0$ for all $i \in [n]$, then $K = \infty$. This result is anticipated since, in this case, the success probability is zero for all clients, and thus the server never receives stochastic gradients.

3.3. Time Complexity

The following theorem gives time complexity for MindFlayer SGD.

Theorem 5 (Proof in Appendix G.2) *Assume that Assumptions 1, 2 and 3 hold. Let $B = \sum_{i=1}^n p_i B_i$ and $\gamma = \frac{1}{2L} \min\left\{1, \frac{\varepsilon B}{\sigma^2}\right\}$ in Method 1. Let $t = (t_1, \ldots, t_n)$, $t_1, \ldots, t_n \geq 0$. Without loss of generality assume that $0 < \tau_1 + t_1 \leq \cdots \leq \tau_n + t_n$. Let*

$$t(m) = \left(\sum_{j=1}^m \frac{p_j}{\tau_j + t_j}\right)^{-1} \left(S + \sum_{j=1}^m p_j\right),$$

where $S = \max\left\{1, \frac{\sigma^2}{\varepsilon}\right\}$. *Let* $m^* = \arg\min_{m\in[n]} t(m)$, *if there are several minimizers we take the smallest one. Put*

$$B_i = \lceil b_i \rceil, \quad b_i = \begin{cases} \frac{t(m^*)}{\tau_i + t_i} - 1, & \text{if } i \leq m^*, \\ 0, & \text{if } i > m^*. \end{cases}$$

Then, MindFlayer SGD *guarantees to find an* ϵ*-stationary point after*

$$T_{\text{MindFlayerSGD}}(t) \geq 8 \times \min_{m\in[n]} \left\{ \left(\frac{1}{m} \sum_{j=1}^{m} \frac{p_j}{\tau_j + t_j} \right)^{-1} \left(\frac{S}{m} + \frac{1}{m} \sum_{j=1}^{m} p_j \right) \frac{\Delta L}{\varepsilon} \right\} \text{ seconds.}$$

The theorem indicates that the optimal strategy is to disregard devices with a high value of $\tau_i + t_i / p_i$. Therefore, we should prioritize devices that not only have a high probability p_i of completing the gradient within the allotted time t_i but also have a relatively small sum of $\tau_i + t_i$. This approach is logical as it avoids including devices with substantial computation times and low probabilities of completing their tasks within the specified duration.

In the deterministic case where $\eta_i = 0$ for all $i \in [n]$, we have $p_i = 1$ for all i. Consequently, by considering the time complexity of MindFlayer SGD with this substitution, the optimal choice of t_i is $t_i = 0$ for all $i \in [n]$. Therefore, the final time complexity becomes

$$T_{\text{MindFlayerSGD}}(t) \geq 8 \times \min_{m\in[n]} \left\{ \left(\frac{1}{m} \sum_{j=1}^{m} \frac{1}{\tau_j} \right)^{-1} \left(\frac{1}{m} + 1 \right) \frac{\Delta L}{\varepsilon} \right\}.$$

This formulation recovers the time complexity for Rennala SGD.

We still have the freedom to choose the t_i allocation times. The optimal strategy would be to select them in a manner that minimizes the time complexity. As observed in Figure 2, setting $t_i = \text{Med}\left[\eta_i\right]$ proves to be a viable choice. This is further confirmed by our experiments in Appendix I.

3.4. Comparing to Rennala SGD

Let $\mathcal{B} := \left\{ (B_1, B_2, \ldots, B_n) : B_i \in \mathbb{N}_0; \sum_{i=1}^{n} B_i = B \right\}$ be the set of all possible batch sizes for each device, the time T_B required for one step with batch size B of Rennala SGD is given by:

$$T_B = \min_{\mathcal{B}} \left\{ \max_{i\in[1,n]} \left\{ B_i \tau_i + \sum_{j=1}^{B_i} \eta_i^j \right\} \right\} \geq T_1 = \min_{i\in[n]} \left\{ \tau_i + \eta_i^1 \right\} \geq \min_{i\in[n]} \left\{ \tau_i \right\} + \min_{i\in[n]} \left\{ \eta_i^1 \right\}.$$

Thus, $\mathbb{E}\left[T_B\right] \geq \tau_{\min} + \mathbb{E}\left[\min_{i\in[n]} \eta_i\right]$,
and the expected time complexity is

$$T_{\text{RennalaSGD}} \geq 8 \times \left(\tau_{\min} + \mathbb{E}\left[\min_{i\in[n]} \eta_i\right]\right) \max\left\{1, \frac{2\sigma^2}{\varepsilon B}\right\} \frac{\Delta L}{\varepsilon}.$$

Note that if the distribution of $\min_{i\in[n]} \eta_i$ is heavy-tailed, then the expected time complexity becomes infinite, thus favoring MindFlayer SGD over Rennala SGD. A simple illustration of this occurs when extending the Equation (2) case, where η is either zero or infinite, to scenarios involving multiple devices. In such cases, the expectation of the minimum time across devices, $\min_{i\in[n]} \eta_i$, also results in an infinite expected time complexity.

Acknowledgements

The research reported in this publication was supported by funding from King Abdullah University of Science and Technology (KAUST): i) KAUST Baseline Research Scheme, ii) Center of Excellence for Generative AI, under award number 5940, iii) SDAIA-KAUST Center of Excellence in Artificial Intelligence and Data Science.

References

[1] Yossi Arjevani, Ohad Shamir, and Nathan Srebro. A tight convergence analysis for stochastic gradient descent with delayed updates. In *Algorithmic Learning Theory*, pages 111–132. PMLR, 2020.

[2] Arda Aytekin, Hamid Reza Feyzmahdavian, and Mikael Johansson. Analysis and implementation of an asynchronous optimization algorithm for the parameter server. *arXiv preprint arXiv:1610.05507*, 2016.

[3] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[4] Alon Cohen, Amit Daniely, Yoel Drori, Tomer Koren, and Mariano Schain. Asynchronous stochastic optimization robust to arbitrary delays. *Advances in Neural Information Processing Systems*, 34:9024–9035, 2021.

[5] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. *Advances in Neural Information Processing Systems*, 24, 2011.

[6] Sanghamitra Dutta, Gauri Joshi, Soumyadip Ghosh, Parijat Dube, and Priya Nagpurkar. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD. In *International Conference on Artificial Intelligence and Statistics*, pages 803–812. PMLR, 2018.

[7] Hamid Reza Feyzmahdavian, Arda Aytekin, and Mikael Johansson. An asynchronous mini-batch algorithm for regularized stochastic optimization. *IEEE Transactions on Automatic Control*, 61(12):3740–3754, 2016.

[8] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[9] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pages 680–690. PMLR, 2020.

[10] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR, 2019.

[11] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[12] Serge Kas Hanna, Rawad Bitar, Parimal Parag, Venkat Dasari, and Salim El Rouayheb. Adaptive distributed stochastic gradient descent for minimizing delay in the presence of stragglers. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4262–4266, 2020. doi: 10.1109/ICASSP40776.2020.9053961.

[13] Serge Kas Hanna, Rawad Bitar, Parimal Parag, Venkat Dasari, and Salim El Rouayheb. Adaptive stochastic gradient descent for fast and communication-efficient distributed learning, 2022.

[14] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[15] Ahmed Khaled and Peter Richtárik. Better theory for SGD in the nonconvex world. *arXiv preprint arXiv:2002.03329*, 2020.

[16] Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Sharper convergence guarantees for asynchronous SGD for distributed and federated learning. *arXiv preprint arXiv:2206.08307*, 2022.

[17] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[18] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. Mnist handwritten digit database, 1998. URL http://yann.lecun.com/exdb/mnist/.

[19] Konstantin Mishchenko, Franck Iutzeler, Jérôme Malick, and Massih-Reza Amini. A delay-tolerant proximal-gradient algorithm for distributed learning. In *International Conference on Machine Learning*, pages 3587–3595. PMLR, 2018.

[20] Konstantin Mishchenko, Francis Bach, Mathieu Even, and Blake Woodworth. Asynchronous SGD beats minibatch SGD under arbitrary delays. *arXiv preprint arXiv:2206.07638*, 2022.

[21] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, pages 3581–3607. PMLR, 2022.

[22] Lam Nguyen, Phuong Ha Nguyen, Marten Dijk, Peter Richtárik, Katya Scheinberg, and Martin Takác. SGD and hogwild! convergence without the bounded gradients assumption. In *International Conference on Machine Learning*, pages 3750–3758. PMLR, 2018.

[23] 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, 2011.

[24] Alexander Tyurin and Peter Richtárik. Optimal time complexities of parallel stochastic optimization methods under a fixed computation model. *Advances in Neural Information Processing Systems*, 36, 2024.

[25] Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. Is local SGD better than minibatch SGD? In *International Conference on Machine Learning*, pages 10334–10343. PMLR, 2020.

[26] Xuyang Wu, Sindri Magnusson, Hamid Reza Feyzmahdavian, and Mikael Johansson. Delay-adaptive step-sizes for asynchronous learning. *arXiv preprint arXiv:2202.08550*, 2022.

Contents

1 Introduction **1**
1.1 Parallel Methods . . . 1

2 Problem Setup and Contributions **2**

3 MindFlayer SGD **4**
3.1 Assumptions . . . 5
3.2 Convergence theory . . . 5
3.3 Time Complexity . . . 5
3.4 Comparing to Rennala SGD . . . 6

A Motivation and Single Device Case **11**

B Related Work **13**

C Conclusion and Future Work **13**

D Heterogeneous Regime **13**
D.1 Related work and discussion . . . 13
D.2 Vecna SGD . . . 14
D.3 Result . . . 15

E The Rennala Algorithm **16**

F The classical SGD theory **16**

G Proofs for Homogeneous Regime **17**
G.1 Proof of Theorem 4 . . . 17
G.1.1 Proof of Theorem 12 . . . 18
G.1.2 Proof of Theorem 13 . . . 19
G.2 Proof of Theorem 5 . . . 20
G.2.1 Proof of Theorem 17 . . . 22

H Proofs for Heterogeneous Regime **23**
H.1 Proof of Theorem 8 . . . 23
H.1.1 Proof of Theorem 15 . . . 24
H.1.2 Proof of Theorem 16 . . . 24
H.2 Proof of Theorem 9 . . . 25
H.2.1 Proof of Theorem 17 . . . 27

I Experiments **27**

Appendix A. Motivation and Single Device Case

To motivate our new method, consider a single device setup as described in Equation (2), where the device either returns a gradient after τ time or fails with probability q. The optimal strategy here is to wait exactly τ seconds. If no gradient is received within this time, it means none will be, so we discard the computation and request a new gradient. Since the probability of getting stuck decreases, we will eventually receive a gradient and progress.

More generally, consider the following two strategies for each step:

- **Strategy 1:** Rennala SGD. We wait for the first B stochastic gradients. Thus, the time for one step for this strategy is the random variable:

$$T_B = \sum_{j=1}^{B}(\tau + \eta^j).$$

- **Strategy 2:** MindFlayer SGD. We repeat the following random trial B times: allocate time t for computing a stochastic gradient. If we do not receive a stochastic gradient within that time, discard the current computation and start over. Then the time for the j-th trial is given by:

$$T^j(t) = \begin{cases} \tau + \eta^j, & \text{if } \eta^j \leq t, \\ \tau + t, & \text{if } \eta^j > t. \end{cases} \tag{3}$$

  Thus, the time for one step for this strategy is the random variable:

$$\tilde{T}_B(t) = \sum_{j=1}^{B} T^j(t).$$

In the second case, rather than waiting for B gradients, we attempt to compute B gradients. Essentially, we limit the time spent on computing a stochastic gradient. In expectation, Strategy 2 will collect Bp gradients per iteration, where $p = P(\eta \leq t)$ is the probability of collecting a gradient within a trial. Setting $t = \infty$ removes this restriction, resulting in the same strategy as the first one, Rennala SGD.

Consider the iteration complexity of Rennala SGD. Assume it takes K iterations. For MindFlayer SGD, each iteration, on average, receives only Bp gradients, making it akin to a scaled-down version of Rennala SGD. Consequently, MindFlayer SGD is expected to require $1/p$ times more iterations than Rennala SGD to achieve the same level of convergence.

Thus, the time complexities in this setting are given by:

$$T_{\text{RennalaSGD}} = K\mathbb{E}\left[T_B\right] = KB(\tau + \mathbb{E}\left[\eta\right]),$$
$$T_{\text{MindFlayerSGD}}(t) = \frac{K}{p}\mathbb{E}\left[\tilde{T}_B(t)\right] = \frac{K}{p}B(\tau + (1-p)t + p\mathbb{E}\left[\tau | \tau \leq t\right]) \leq \frac{K}{p}B(\tau + t).$$

This leads us to the following proposition:

Proposition 6 *For the $n = 1$ case,* MindFlayer SGD *is faster than* Rennala SGD *if there exist a time threshold $t > 0$ for which the following holds*

$$\frac{\tau + t}{P(\eta \leq t)} < \tau + \mathbb{E}\left[\eta\right].$$

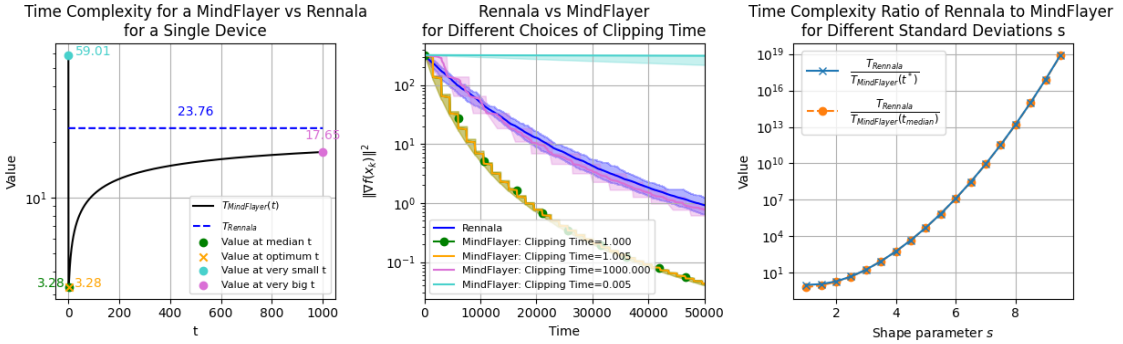

Figure 2: **Left:** We compare the time complexity of MindFlayer SGD as a function of clipping time (t) to the constant complexity of Rennala SGD, highlighting the adaptive efficiency of MindFlayer SGD across different t values. **Middle:** Empirical validation shows consistent performance improvements for MindFlayer SGD using the same clipping times as in the left graph. **Right:** The ratio of time complexities between Rennala SGD and MindFlayer SGD is plotted against different standard deviations (s), demonstrating exponential efficiency gains for MindFlayer SGD at optimal clipping times, with similar trends at median clipping times.

Note that this can hold for a big range of values t and even for any finite value t. The latter holds for example in the case when $\mathbb{E}[\eta] = \infty$. An example of such a scenario was demonstrated in Equation (2). There are many other distributions for which the expectation is not finite, such as the Log-Cauchy distribution, Lévy distribution, Log-t distribution, Landau distribution, and so forth.

A less restrictive example of distributions are positively-skewed distributions. Let $s = \mathbb{E}[\eta] - \text{Med}[\eta]$ be the skewness coefficient of the distribution $\mathcal{J}$. If $s > 0$ we say that the distribution is positively-skewed. Then we have the following proposition.

Proposition 7 *For the $n = 1$ case, if $s > \tau + \text{Med}[\eta]$ then* MindFlayer SGD *is faster than* Rennala SGD. *Moreover, if $s = (\tau + \text{Med}[\eta])(2\alpha - 1)$ then*

$$\frac{T_{\text{RennalaSGD}}}{T_{\text{MindFlayerSGD}}(\text{Med}[\eta])} \geq \alpha.$$

Therefore, Rennala SGD can be arbitrarily bad. As an example consider the Lognormal(μ, σ^2) distribution. For this distribution, we have:

$$s = \mathbb{E}[\eta] - \text{Med}[\eta] = \exp\left(\mu + \frac{\sigma^2}{2}\right) - \exp(\mu).$$

Thus, as we increase σ, the difference becomes arbitrarily large.

To verify this, we also conducted a small experiment, see Figure 2. The right plot showcases how the ratio of time complexity between Rennala SGD and MindFlayer SGD can get arbitrarily large for the optimal clipping time $t^* := \arg\min_t T_{\text{MindFlayerSGD}}(t)$ and even the median of the distribution $t_{\text{median}} = \text{Med}[\eta]$. The left and middle plots showcase the potential improvement, and even loss from choosing the clipping times t.

Appendix B. Related Work

There are several other related works. Dutta et al. [6] explore the error-runtime trade-offs in Distributed SGD, revealing how slower and stale gradients can sometimes enhance convergence processes. Woodworth et al. [25] compare Local SGD with Minibatch SGD, analyzing the efficiency of local updates in different distributed settings. Wu et al. [26] advance the understanding of asynchronous methods by proposing delay-adaptive step-sizes that adjust to asynchronous learning environments, optimizing the convergence rates. Furthermore, Hanna et al. [12, 13] focus on adaptive stochastic gradient descent to improve communication efficiency in distributed learning, offering strategies that reduce communication demands while maintaining fast convergence.

Appendix C. Conclusion and Future Work

In this paper, we tackle the problem of minimizing nonconvex functions with Lipschitz gradients using parallel workers for stochastic gradient computation. We focus on the challenging scenario where worker compute times are heterogeneous and random, building on recent ASGD methods like Rennala SGD. While Rennala SGD is effective under deterministic conditions, its performance deteriorates with random compute times.

To address this, we propose a novel method called MindFlayer SGD, which adapts to computation time variability by discarding and restarting delayed computations, rather than adhering to fixed batch sizes. This approach enhances robustness, allowing MindFlayer SGD to outperform both Rennala SGD and standard ASGD in theoretical and empirical tests, especially in environments with high variance and heavy-tailed compute time distributions.

Our results show significant reductions in time complexity, validated through simulations under diverse conditions. Future work will incorporate communication times and explore the use of gradient estimators with varying variance bounds to further improve optimization performance in federated learning settings.

Appendix D. Heterogeneous Regime

Up to this point, we discussed the regime when all workers calculate i.i.d. stochastic gradients. In distributed optimization and federated learning [17], it can be possible that the workers hold different datasets. Let us consider the following optimization problem:

$$\min_{x\in\mathbb{R}^d}\left\{f(x) := \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left[f_i(x;\xi_i)\right]\right\}, \tag{4}$$

where $f_i : \mathbb{R}^d \times \mathbb{S}_i \to \mathbb{R}^d$ and ξ_i are random variables with some distributions $\mathcal{D}_i$ on $\mathbb{S}_i$. Problem (4) generalizes problem (**??**). Here we have the same goal as in the previous sections.

D.1. Related work and discussion

The optimization problem (4) is well-investigated by many papers, including [2, 16, 19–21, 26]. There were attempts to analyze Asynchronous SGD in the heterogeneous regime. For instance, Mishchenko et al. [20] proved the convergence to a neighborhood of a solution only. In general, it is quite challenging to get good rates for Asynchronous SGD without additional assumptions about the similarity of the functions f_i [16, 20].

In the non-stochastic case, when $\sigma^2 = 0$, Wu et al. [26] analyzed the PIAG method in the non-stochastic heterogeneous regime and showed convergence. Although the performance of PIAG can be good in practice, in the worst case PIAG requires $\mathrm{O}\left(\tau_n \widehat{L}\Delta/\varepsilon\right)$ seconds to converge, where τ_n is the time delay of the slowest worker, $\widehat{L} := \sqrt{\sum_{i=1}^n L_i^2}$, and L_i is a Lipschitz constant of ∇f_i. Note that the synchronous Minibatch SGD (see Section **??**) method has the complexity $\mathrm{O}\left(\tau_n L\Delta/\varepsilon\right)$, which is always better.[3]

Our lower bound in Theorem **??** does not leave hope of breaking the dependence on the worst straggler in the heterogeneous case. In the stochastic case, the lower bound is slightly more optimistic in the regimes when the statistical term (the second term in (**??**)) is large. If the stragglers do not have too large delays, then their contributions to the *arithmetic* mean can be small. Note that in Theorem **??** in the homogeneous case, we have the *harmonic* mean of the delays instead.

D.2. Vecna SGD

Here we describe our method called Vecna SGD [4].

Algorithm 3 Vecna SGD

Input: starting point $x^0 \in \mathbb{R}^d$, stepsize $\gamma > 0$, allotted times $t_1, \ldots, t_n \geq 0$, number of trials per client $B_1, \ldots, B_n \geq 0$
for $k = 1, 2, \ldots, K$ **do**
  Put $g_i^k = 0$
  Send x^k to all clients
  Run Method 4 in all clients $i = 1, 2, \ldots, n$
  while there is a client that has trials to perform **do**
    Wait for the fastest client
    Receive gradient g_i from client i
    $g_i^k = g_i^k + g$
  end while
  $g^k = \frac{1}{n}\sum_{i=1}^n \frac{g_i^k}{p_i B_i}$, ◇ $p_i = F_i(t_i) = P(\eta_i \leq t_i)$.
  $x^{k+1} = x^k - \gamma g^k$
end for

Algorithm 4 Client i-s k-th step

Receive x^k from the server
for $j = 1, 2, \ldots, B_i$ **do**
  Sample $\eta_i^j \sim \mathcal{J}_i$ ◇ Start computing gradient estimator.
  if $\tau_i + \eta_i^j \leq t_i$ **then**
    $g = \nabla f(x^k; \xi_i^j), \ \xi_i^j \sim \mathcal{D}$ ◇ The computation completes within the allotted time t_i.
    Send g to the server
  end if
end for

3. In the nonconvex case, $\widehat{L}$ can be arbitrarily larger than L.
4. We name our method Vecna SGD, drawing inspiration from Vecna from *Stranger Things*.

The MindFlayer SGD algorithm begins with an initialization at a starting point x^0 in $\mathbb{R}^d$, with a specified stepsize γ, time allowances t_i, and trial counts B_i for each client. In each iteration k, ranging from $k = 1$ to K, the server distributes the current point x^k to all clients. Each client i then executes a subroutine (Method 2) to attempt to compute B_i stochastic gradients from samples ξ_i^j drawn from a distribution $\mathcal{D}$. During each attempt, client i starts computing a stochastic gradient; if the computation exceeds the allotted time t_i, they discard the current gradient and begin another computation. Consequently, the actual number of stochastic gradients received from each client i becomes a random variable, ranging from 0 to B_i. The expected number of gradients from client i is given by $p_i B_i$, leading to an overall expected total of stochastic gradients $B = \sum_{i=1}^n p_i B_i$. The server aggregates these received stochastic gradients and normalizes the collective gradient by the expected batch size B. Finally, the point is updated to $x^{k+1} = x^k - \gamma g^k$ following each aggregation round.

D.3. Result

Theorem 8 (Proof in Appendix H.1) *Assume that Assumptions 1, 2 hold for the function f and Assumption 3 holds for the function f_i for all $i \in [n]$. Let $\gamma = \min\left\{\frac{1}{\sqrt{LAK}}, \frac{1}{LB}, \frac{\varepsilon}{2LC}\right\}$ in Method 3. Then after*

$$K \geq \frac{12\Delta L}{\varepsilon} \max\left\{B, \frac{12\Delta A}{\varepsilon}, \frac{2C}{\varepsilon}\right\},$$

iterations, the method guarantees that $\min_{0\leq k\leq K} \mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2\right] \leq \varepsilon$, where

$$A = \frac{L}{n^2}\sum_{i=1}^n \frac{1-p_i}{p_i B_i}, \quad B = 1, \quad C = \frac{\sigma^2}{n^2}\sum_{i=1}^n \frac{1}{p_i B_i}.$$

Theorem 9 (Proof in Appendix H.2) *Assume that Assumptions 1, 2 hold for the function f and Assumption 3 holds for the function f_i for all $i \in [n]$. Let $\gamma = \min\left\{\frac{1}{\sqrt{LAK}}, \frac{1}{LB}, \frac{\varepsilon}{2LC}\right\}$ in Method 3. Let $t = (t_1, \ldots, t_n)$, $t_1, \ldots, t_n \geq 0$. Without loss of generality assume that $0 < \tau_1 + t_1 \leq \cdots \leq \tau_n + t_n$. Let*

$$T = \tau_n + t_n + \left[\frac{1}{n}\sum_{i=1}^n \frac{\tau_i + t_i}{p_i}\right]\frac{\sigma^2}{n\varepsilon} + \left[\frac{1}{n}\sum_{i=1}^n \frac{1-p_i}{p_i}(\tau_i + t_i)\right]\frac{\Delta L}{n\varepsilon}.$$

Put

$$B_i = \lceil b_i \rceil, \quad b_i = \frac{T}{\tau_i + t_i}.$$

Then, Vecna SGD guarantees to find an ϵ-stationary point after

$$T_{\text{VecnaSGD}}(t) \geq 288 \times \frac{\Delta L}{\varepsilon}\left(\tau_n + t_n + \left[\frac{1}{n}\sum_{i=1}^n \frac{\tau_i + t_i}{p_i}\right]\frac{\sigma^2}{n\varepsilon} + \left[\frac{1}{n}\sum_{i=1}^n \frac{1-p_i}{p_i}(\tau_i + t_i)\right]\frac{\Delta L}{n\varepsilon}\right)$$

seconds.

Appendix E. The Rennala Algorithm

Algorithm 5 Rennala SGD

Input: starting point x^0, stepsize γ, batch size S
Run Method 6 in all workers
for $k = 0, 1, \ldots, K-1$ **do**
  Init $g^k = 0$ and $s = 1$
  while $s \leq S$ **do**
    Wait for the next worker
    Receive gradient and iteration index (g, k')
    if $k' = k$ **then**
      $g^k = g^k + \frac{1}{S} g; \quad s = s + 1$
    end if
    Send (x^k, k) to the worker
  end while
  $x^{k+1} = x^k - \gamma g^k$
end for

Algorithm 6 Worker's Infinite Loop

Init $g = 0$ and $k' = -1$
while True **do**
  Send (g, k') to the server
  Receive (x^k, k) from the server
  $k' = k$
  $g = \nabla f(x^k; \xi), \quad \xi \sim \mathcal{D}$
end while

We mention the Rennala SGD throughout the paper, here we provide a brief introduction to the method and its development. Algorithm Algorithm 5 shows the work done by the server. Essentially, the server asynchronously waits to collect a batch of size S, whenever it recieves a gradient from a worker that has the same iteration as the algorithm, it assigns it to compute a gradient at the same point x_k. After collecting the batch, we preform a synchronous update (given that all gradients were made on the same point x_k), using an average of the collected batch.

Appendix F. The classical SGD theory

In this section, we present the classical SGD theory as developed by Ghadimi and Lan [8] and Khaled and Richtárik [15]. Our analysis will follow the approach of the latter.

We consider the stochastic gradient descent (SGD) method:

$$x^{k+1} = x^k - \gamma g(x^k),$$

where $x^0 \in \mathbb{R}^d$ is the initial point, and $g(x)$ is a stochastic gradient estimator at x.

We make the following assumption:

Assumption 10 *The stochastic gradient estimator $g(x)$ satisfies:*

$$\mathbb{E}\left[g(x)\right] = \nabla f(x)$$
$$\mathbb{E}\left[\|g(x)\|^2\right] \leq 2A\left(f(x) - f^*\right) + B\|\nabla f(x)\|^2 + C,$$

for all $x \in \mathbb{R}^d$ and some constants $A, B, C \geq 0$.

This assumption is both general and reasonable, and it is satisfied by many modern SGD-type methods. For further details, refer to Khaled and Richtárik [15].

Under this assumption, we can derive the following convergence result.

Theorem 11 (Corollary 1 [15]) *Assume that Assumptions 1, 2 and 10 hold. Then for any* $\varepsilon > 0$

$$\min_{0\leq k\leq K} \mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2\right] \leq \varepsilon$$

for

$$\gamma = \min\left\{\frac{1}{\sqrt{LAK}}, \frac{1}{LB}, \frac{\varepsilon}{2LC}\right\},$$

and

$$K \geq \frac{12\Delta L}{\varepsilon}\max\left\{B, \frac{12\Delta A}{\varepsilon}, \frac{2C}{\varepsilon}\right\}.$$

Appendix G. Proofs for Homogeneous Regime

G.1. Proof of Theorem 4

First, we rewrite MindFlayer SGD in a classical SGD way where we do gradient step with an unbiased estimator of the gradient at each iteration.

Algorithm 7 MindFlayer SGD

Input: starting point x^0, stepsize γ, time budgets $t_1, \ldots, t_n \geq 0$, batch sizes $B_1, \ldots, B_n \geq 0$,
for $k = 0, 1, \ldots, K-1$ **do**
  $g^k = \frac{1}{B}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right)\nabla f\left(x^k; \xi_i^j\right)$
  $x^{k+1} = x^k - \gamma g^k$
end for

where $B = \sum_{i=1}^n p_i B_i$, $p_i = F(t_i) = P(\eta_i \leq t_i)$ and $I(\cdot)$ denotes the indicator function. To prove the theorem we need to establish some properties of the gradient estimator. First, we need an unbiased estimator.

Lemma 12 (Proof in ?? G.1.1) *The gradient estimator in Algorithm 7 given by*

$$g(x) := \frac{1}{B}\sum_{i=1}^{n}\sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right)\nabla f\left(x; \xi_i^j\right)$$

is unbiased, i.e., $\mathbb{E}\left[g(x)\right] = \nabla f(x)$ *for all* $x \in \mathbb{R}^d$.

Next, we obtain an upper bound for the variance of this estimator.

Lemma 13 (Proof in ?? G.1.2) *The gradient estimator in Algorithm 7 given by*

$$g(x) := \frac{1}{B}\sum_{i=1}^{n}\sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right)\nabla f\left(x; \xi_i^j\right)$$

satisfies

$$\mathbb{E}\left[\left\|g(x)^2\right\|\right] \leq 2\left\|\nabla f(x)\right\|^2 + \frac{1}{B}\sigma^2.$$

We are ready to prove the Theorem 4.

Theorem 4 *Assume that Assumptions 1, 2 and 3 hold. Let* $B = \sum_{i=1}^n p_i B_i$ *and* $\gamma = \frac{1}{2L}\min\left\{1, \frac{\varepsilon B}{\sigma^2}\right\}$ *in Method 1. Letting* $\Delta := f(x^0) - f^*$*, we have that after*

$$K \geq \max\left\{1, \frac{\sigma^2}{\varepsilon B}\right\}\frac{8\Delta L}{\varepsilon}$$

iterations, the method guarantees that $\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2\right] \leq \varepsilon$.

Proof Note that Algorithm 1 can be viewed as a special case of classical stochastic gradient descent (SGD), as reformulated in Algorithm 7. We need to verify that the gradient estimator fulfills the conditions required by classical SGD (Theorem 11). The two preceding lemmas address this requirement precisely. Specifically, Theorem 12 confirms that the gradient estimator used in Algorithm 7 is unbiased, while Theorem 13 verifies that the variance of this estimator meets the conditions specified in Assumption 10, with $A = 0$, $B = 2$ and $C = \frac{\sigma^2}{B}$. Consequently, it remains to apply Theorem 11. ■

G.1.1. PROOF OF THEOREM 12

Lemma 12 *The gradient estimator in Algorithm 7 given by*

$$g(x) := \frac{1}{B}\sum_{i=1}^n\sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right)\nabla f\left(x;\xi_i^j\right)$$

is unbiased, i.e., $\mathbb{E}\left[g(x)\right] = \nabla f(x)$ *for all* $x \in \mathbb{R}^d$*, where* $B = \sum_{i=1}^n p_i B_i$.

Proof This follows from direct computation:

$$\begin{aligned}
\mathbb{E}\left[g(x)\right] &= \mathbb{E}\left[\frac{1}{B}\sum_{i=1}^n\sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right)\nabla f\left(x;\xi_i^j\right)\right] \\
&= \frac{1}{B}\sum_{i=1}^n\sum_{j=1}^{B_i}\mathbb{E}\left[I\left(\eta_i^j \leq t_i\right)\nabla f\left(x;\xi_i^j\right)\right] \\
&\overset{(\eta_i^j \perp\!\!\!\perp \xi_i^j)}{=} \frac{1}{B}\sum_{i=1}^n\sum_{j=1}^{B_i}\mathbb{E}\left[I\left(\eta_i^j \leq t_i\right)\right]\mathbb{E}\left[\nabla f\left(x;\xi_i^j\right)\right] \\
&= \frac{1}{B}\sum_{i=1}^n\sum_{j=1}^{B_i} p_i\nabla f(x) \\
&= \nabla f(x)\frac{1}{B}\sum_{i=1}^n p_i B_i \\
&= \nabla f(x).
\end{aligned}$$

■

G.1.2. PROOF OF THEOREM 13

Lemma 13 *The gradient estimator in Algorithm 7 given by*

$$g(x) := \frac{1}{B}\sum_{i=1}^{n}\sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right)\nabla f\left(x;\xi_i^j\right)$$

satisfies

$$\mathbb{E}\left[\left\|g(x)^2\right\|\right] \leq 2\left\|\nabla f(x)\right\|^2 + \frac{1}{B}\sigma^2,$$

where $B = \sum_{i=1}^{n} p_i B_i$.

Proof In order to simplify notation, let

$$a_i := \sum_{j=1}^{B_i} b_i^j,$$

where

$$b_i^j := I\left(\eta_i^j \leq t_i\right)\nabla f\left(x;\xi_i^j\right).$$

Step 1 (Initial expression). We express $\mathbb{E}\left[\|g(x)\|^2\right]$ in terms of a_i:

$$\mathbb{E}\left[\|g(x)\|^2\right] = \mathbb{E}\left[\left\|\frac{1}{B}\sum_{i=1}^{n} a_i\right\|^2\right] = \frac{1}{B^2}\mathbb{E}\left[\sum_{i=1}^{n}\|a_i\|^2 + \sum_{i\neq j}\langle a_i, a_j\rangle\right].$$

We further simplify both terms via:

$$\|a_i\|^2 = \left\|\sum_{j=1}^{B_i} b_i^j\right\|^2 = \sum_{j=1}^{B_i}\left\|b_i^j\right\|^2 + \sum_{k\neq l}\left\langle b_i^k, b_i^l\right\rangle, \tag{5}$$

$$\langle a_i, a_j\rangle = \left\langle\sum_{k=1}^{B_i} b_i^k, \sum_{l=1}^{B_j} b_j^l\right\rangle = \sum_{k=1}^{B_i}\sum_{l=1}^{B_j}\left\langle b_i^k, b_j^l\right\rangle. \tag{6}$$

Step 2. (Finding the expectations). Further

$$\begin{aligned}
\mathbb{E}\left[\left\|b_i^j\right\|^2\right] &= \mathbb{E}\left[\left(I\left(\eta_i^j \leq t_i\right)\right)^2\left\|\nabla f\left(x;\xi_i^j\right)\right\|^2\right] \\
&\overset{(\eta_i^j \perp\!\!\!\perp \xi_i^j)}{=} \mathbb{E}\left[\left(I\left(\eta_i^j \leq t_i\right)\right)^2\right]\mathbb{E}\left[\left\|\nabla f\left(x;\xi_i^j\right)\right\|^2\right] \\
&\leq p_i\left(\|\nabla f(x)\|^2 + \mathbb{E}\left[\left\|\nabla f\left(x;\xi_i^j\right) - \nabla f(x)\right\|^2\right]\right) \\
&\overset{(Assumption\ 3)}{\leq} p_i\left(\|\nabla f(x)\|^2 + \sigma^2\right),
\end{aligned} \tag{7}$$

and

$$\begin{aligned}\mathbb{E}\left[\left\langle b_i^k, b_j^l\right\rangle\right] &= \mathbb{E}\left[\left\langle I\left(\eta_i^k \le t_i\right)\nabla f\left(x;\xi_i^k\right), I\left(\eta_j^l \le t_j\right)\nabla f\left(x;\xi_j^l\right)\right\rangle\right] \\ &\overset{(\perp)}{=} \mathbb{E}\left[I\left(\eta_i^k \le t_i\right)\right]\mathbb{E}\left[I\left(\eta_j^l \le t_j\right)\right]\left\langle \mathbb{E}\left[\nabla f\left(x;\xi_i^k\right)\right], \mathbb{E}\left[\nabla f\left(x;\xi_j^l\right)\right]\right\rangle \\ &= p_i p_j \left\|\nabla f(x)\right\|^2. \end{aligned} \tag{8}$$

Step 3 (Putting everything together). We start with

$$\begin{aligned}\mathbb{E}\left[\left\|a_i\right\|^2\right] &\overset{(5,7,8)}{\le} B_i p_i\left(\left\|\nabla f(x)\right\|^2 + \sigma^2\right) + B_i\left(B_i - 1\right)p_i^2\left\|\nabla f(x)\right\|^2 \\ &\le B_i p_i\left(\left\|\nabla f(x)\right\|^2 + \sigma^2\right) + B_i^2 p_i^2\left\|\nabla f(x)\right\|^2,\end{aligned}$$

using this and recalling the definition of B, we get

$$\mathbb{E}\left[\sum_{i=1}^n \left\|a_i\right\|^2\right] \le B\left\|\nabla f(x)\right\|^2 + B\sigma^2 + \left\|\nabla f(x)\right\|^2 \sum_{i=1}^n B_i^2 p_i^2.$$

Next

$$\left\langle a_i, a_j\right\rangle \overset{(6,8)}{=} B_i p_i B_j p_j \left\|\nabla f(x)\right\|^2,$$

finally,

$$\begin{aligned}\mathbb{E}\left[\left\|g(x)\right\|^2\right] &= \frac{1}{B^2}\mathbb{E}\left[\sum_{i=1}^n \left\|a_i\right\|^2 + \sum_{i\ne j}\left\langle a_i, a_j\right\rangle\right] \\ &\le \frac{1}{B^2}\left[B\left\|\nabla f(x)\right\|^2 + B\sigma^2 + \left(\sum_{i=1}^n B_i^2 p_i^2 + \sum_{i\ne j} B_i p_i B_j p_j\right)\left\|\nabla f(x)\right\|^2\right] \\ &= \frac{1}{B^2}\left(B + B^2\right)\left\|\nabla f(x)\right\|^2 + \frac{\sigma^2}{B} \\ &\le 2\left\|\nabla f(x)\right\|^2 + \frac{\sigma^2}{B}.\end{aligned}$$

■

G.2. Proof of Theorem 5

The following lemma gives time complexity for any choice of $B_1, \ldots, B_n$ and $t = (t_1, \ldots, t_n)$ in MindFlayer SGD.

Lemma 14 (Proof in ?? H.2.1) *Assume that Assumptions 1, 2 and 3 hold. Let $B = \sum_{i=1}^n p_i B_i$ and $\gamma = \frac{1}{2L}\min\left\{1, \frac{\varepsilon B}{\sigma^2}\right\}$ in Method 1. Then after*

$$T_{\text{MindFlayerSGD}}(t) \ge \max_{i\in[n]}\left\{B_i\left(\tau_i + t_i\right)\right\}\max\left\{1, \frac{\sigma^2}{\varepsilon B}\right\}\frac{8\Delta L}{\varepsilon}$$

seconds, the method guarantees to find an ϵ-stationary point.

Now we are ready to prove the theorem.

Theorem 5 *Assume that Assumptions 1, 2 and 3 hold. Let* $B = \sum_{i=1}^n p_i B_i$ *and* $\gamma = \frac{1}{2L}\min\left\{1, \frac{\varepsilon B}{\sigma^2}\right\}$ *in Method 1. Let* $t = (t_1, \ldots, t_n)$, $t_1, \ldots, t_n \geq 0$. *Without loss of generality assume that* $0 < \tau_1 + t_1 \leq \cdots \leq \tau_n + t_n$. *Let*

$$t(m) = \left(\sum_{j=1}^m \frac{p_j}{\tau_j + t_j}\right)^{-1}\left(S + \sum_{j=1}^m p_j\right),$$

where $S = \max\left\{1, \frac{\sigma^2}{\varepsilon}\right\}$. *Let* $m^* = \arg\min_{m\in[n]} t(m)$, *if there are several minimizers we take the smallest one. Put*

$$B_i = \lceil b_i \rceil, \quad b_i = \begin{cases} \frac{t(m^*)}{\tau_i + t_i} - 1, & \text{if } i \leq m^*, \\ 0, & \text{if } i > m^*. \end{cases}$$

Then, MindFlayer SGD *guarantees to find an* ϵ*-stationary point after*

$$T_{\text{MindFlayerSGD}}(t) \geq 8 \times \min_{m\in[n]}\left\{\left(\frac{1}{m}\sum_{j=1}^m \frac{p_j}{\tau_j + t_j}\right)^{-1}\left(\frac{S}{m} + \frac{1}{m}\sum_{j=1}^m p_j\right)\frac{\Delta L}{\varepsilon}\right\}$$

seconds.

Proof First we show that B_i-s are valid choice, i.e. $b_i > 0$ for $i \leq m^*$. If $m^* = 1$, then $t(1) = \frac{\tau_1 + t_1}{p_1}(S + p_1)$, thus $b_1 = \frac{S}{p_1} > 0$. If $m^* > 1$, then, by its definition, $t(m^*) < t(m^* - 1)$. This implies

$$\left(\sum_{j=1}^{m^*} \frac{p_j}{\tau_j + t_j}\right)^{-1}\left(S + \sum_{j=1}^{m^*} p_j\right) < \left(\sum_{j=1}^{m^*-1} \frac{p_j}{\tau_j + t_j}\right)^{-1}\left(S + \sum_{j=1}^{m^*-1} p_j\right),$$

leading to

$$\left(\sum_{j=1}^{m^*-1} \frac{p_j}{\tau_j + t_j}\right)\left(S + \sum_{j=1}^{m^*} p_j\right) < \left(\sum_{j=1}^{m^*} \frac{p_j}{\tau_j + t_j}\right)\left(S + \sum_{j=1}^{m^*-1} p_j\right)$$

and

$$p_{m^*}\left(\sum_{j=1}^{m^*} \frac{p_j}{\tau_j + t_j}\right) < \frac{p_{m^*}}{\tau_{m^*} + t_{m^*}}\left(S + \sum_{j=1}^{m^*} p_j\right).$$

From the last inequality, we get that $\tau_{m^*} + t_{m^*} < t(m^*)$, thus $b_i \geq b_{m^*} > 0$ for all $i \leq m^*$.

It remains to find the time complexity with these choices of B_i. From Theorem 17, we have that the time complexity is

$$T_{\text{MindFlayerSGD}}(t) \geq \max_{i\in[n]}\{B_i(\tau_i + t_i)\}\max\left\{1, \frac{\sigma^2}{\varepsilon B}\right\}\frac{8\Delta L}{\varepsilon}.$$

Then,

$$\max_{i\in[n]}\{B_i(\tau_i+t_i)\} \leq \max_{i\in[n]}\{(b_i+1)(\tau_i+t_i)\} = t(m^*).$$

On the other hand

$$\begin{aligned} B &= \sum_{i=1}^{n} p_i B_i \geq \sum_{i=1}^{n} p_i b_i = \sum_{i=1}^{m^*} t(m^*)\frac{p_i}{\tau_i+t_i} - \sum_{i=1}^{m^*} p_i \\ &= \left(\sum_{j=1}^{m^*}\frac{p_j}{\tau_j+t_j}\right)^{-1}\left(S+\sum_{j=1}^{m^*}p_j\right)\sum_{i=1}^{m^*}\frac{p_i}{\tau_i+t_i} - \sum_{i=1}^{m^*}p_i = S \geq \frac{\sigma^2}{\varepsilon}. \end{aligned}$$

Therefore, the time complexity is

$$T_{\text{MindFlayerSGD}}(t) \geq t(m^*)\frac{8\Delta L}{\varepsilon} = \frac{8\Delta L}{\varepsilon}\min_{m\in[n]}\left\{\left(\sum_{j=1}^{m}\frac{p_j}{\tau_j+t_j}\right)^{-1}\left(S+\sum_{j=1}^{m}p_j\right)\right\}.$$

■

G.2.1. PROOF OF THEOREM 17

Lemma 17 *Assume that Assumptions 1, 2 and 3 hold. Let $B = \sum_{i=1}^{n} p_i B_i$ and $\gamma = \frac{1}{2L}\min\left\{1, \frac{\varepsilon B}{\sigma^2}\right\}$ in Method 1. Then after*

$$T_{\text{MindFlayerSGD}}(t) \geq \max_{i\in[n]}\{B_i(\tau_i+t_i)\}\max\left\{1, \frac{\sigma^2}{\varepsilon B}\right\}\frac{8\Delta L}{\varepsilon}$$

seconds, the method guarantees to find an ϵ-stationary point.

Proof Let $T_i^j(t_i)$ be the random time taken by client i in the j-th attempt of calculating gradient estimator. We have

$$T_i^j(t_i) = \begin{cases} \tau_i + \eta_i^j, & \text{if } \eta_i^j \leq t_i, \\ \tau_i + t_i, & \text{if } \eta_i^j > t_i. \end{cases} \tag{9}$$

Thus, the random time taken for client i to finish it's all b_i trials is

$$\mathcal{T}_i(t_i) := \sum_{j=1}^{b_i} T_i^j(t_i) \leq b_i(\tau_i+t_i). \tag{10}$$

Finally, let $\mathcal{T}$ be the random time required for one iteration of MindFlayer SGD. We get

$$\mathcal{T} = \max_{i\in[n]}\mathcal{T}_i(t_i) \leq \max_{i\in[n]}\{b_i(\tau_i+t_i)\}. \tag{11}$$

It remains to multiply $\mathcal{T}$ with the number of iterations K given by Theorem 4. ■

Appendix H. Proofs for Heterogeneous Regime

H.1. Proof of Theorem 8

Here, we rewrite Vecna SGD (Algorithm 3) in a classical SGD way.

Algorithm 8 Vecna SGD

Input: starting point x^0, stepsize γ, time budgets $t_1, \ldots, t_n \geq 0$, batch sizes $b_1, \ldots, b_n \geq 0$,
for $k = 0, 1, \ldots, K-1$ **do**
  $g^k = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{p_i B_i} \sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right) \nabla f_i\left(x^k; \xi_i^j\right)$
  $x^{k+1} = x^k - \gamma g^k$
end for

where $p_i = F(t_i) = P(\eta_i \leq t_i)$.

To prove the theorem we need to establish some properties of the gradient estimator. First, we need an unbiased estimator.

Lemma 15 (Proof in ?? H.1.1) *The gradient estimator in Algorithm 8 given by*

$$g(x) := \frac{1}{n} \sum_{i=1}^{n} \frac{1}{p_i B_i} \sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right) \nabla f_i\left(x; \xi_i^j\right)$$

is unbiased, i.e., $\mathbb{E}\left[g(x)\right] = \nabla f(x)$ *for all* $x \in \mathbb{R}^d$.

Next, we obtain an upper bound for the variance of this estimator.

Lemma 16 (Proof in ?? H.1.2) *The gradient estimator in Algorithm 8 given by*

$$g(x) := \frac{1}{n} \sum_{i=1}^{n} \frac{1}{p_i B_i} \sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right) \nabla f_i\left(x; \xi_i^j\right)$$

satisfies

$$\mathbb{E}\left[\left\|g(x)^2\right\|\right] \leq \frac{2L\Delta}{n^2} \sum_{i=1}^{n} \frac{1-p_i}{p_i B_i} + \|\nabla f(x)\|^2 + \frac{\sigma^2}{n^2} \sum_{i=1}^{n} \frac{1}{p_i B_i}.$$

We are ready to prove Theorem 8. First, let us restate the theorem.

Theorem 8 *Assume that Assumptions 1, 2 hold for the function* f *and Assumption 3 holds for the function* f_i *for all* $i \in [n]$. *Let* $\gamma = \min\left\{\frac{1}{\sqrt{LAK}}, \frac{1}{LB}, \frac{\varepsilon}{2LC}\right\}$ *in Method 3. Then after*

$$K \geq \frac{12\Delta L}{\varepsilon} \max\left\{B, \frac{12\Delta A}{\varepsilon}, \frac{2C}{\varepsilon}\right\},$$

iterations, the method guarantees that $\min_{0 \leq k \leq K} \mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2\right] \leq \varepsilon$, *where*

$$A = \frac{L}{n^2} \sum_{i=1}^{n} \frac{1-p_i}{p_i B_i}, \quad B = 1, \quad C = \frac{\sigma^2}{n^2} \sum_{i=1}^{n} \frac{1}{p_i B_i}.$$

Proof Note that Algorithm 3 can be viewed as a special case of classical stochastic gradient descent (SGD), as reformulated in Algorithm 8. We need to verify that the gradient estimator fulfills the conditions required by classical SGD (Theorem 11). The two preceding lemmas address this requirement precisely. Specifically, Theorem 15 confirms that the gradient estimator used in Algorithm 8 is unbiased, while Theorem 16 verifies that the variance of this estimator meets the conditions specified in Assumption 10. Consequently, it remains to apply Theorem 11. ■

H.1.1. PROOF OF THEOREM 15

Lemma H.1.1 *The gradient estimator in Algorithm 8 given by*

$$g(x) := \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i B_i}\sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right)\nabla f_i\left(x;\xi_i^j\right)$$

is unbiased, i.e., $\mathbb{E}\left[g(x)\right] = \nabla f(x)$ *for all* $x \in \mathbb{R}^d$.

Proof This follows from direct computation:

$$\begin{aligned}
\mathbb{E}\left[g(x)\right] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i B_i}\sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right)\nabla f_i\left(x;\xi_i^j\right)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i B_i}\sum_{j=1}^{B_i}\mathbb{E}\left[I\left(\eta_i^j \leq t_i\right)\nabla f_i\left(x;\xi_i^j\right)\right] \\
&\overset{(\eta_i^j \perp \xi_i^j)}{=} \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i B_i}\sum_{j=1}^{B_i}\mathbb{E}\left[I\left(\eta_i^j \leq t_i\right)\right]\mathbb{E}\left[\nabla f_i\left(x;\xi_i^j\right)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i B_i}\sum_{j=1}^{B_i} p_i \nabla f_i(x) \\
&= \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x) \\
&= \nabla f(x).
\end{aligned}$$

■

H.1.2. PROOF OF THEOREM 16

Lemma 16 *The gradient estimator in Algorithm 8 given by*

$$g(x) := \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_i B_i}\sum_{j=1}^{B_i} I\left(\eta_i^j \leq t_i\right)\nabla f_i\left(x;\xi_i^j\right)$$

satisfies

$$\mathbb{E}\left[\left\|g(x)^2\right\|\right] \leq \frac{2L\Delta}{n^2}\sum_{i=1}^{n}\frac{1-p_i}{p_i B_i} + \|\nabla f(x)\|^2 + \frac{\sigma^2}{n^2}\sum_{i=1}^{n}\frac{1}{p_i B_i}.$$

Proof Since η_i^j and ξ_i^j are independent from each other for all $i \in [n]$ and j, we have

$$\operatorname{Var}\left(g(x)\right) = \frac{1}{n^2}\sum_{i=1}^{n}\frac{1}{p_i^2 B_i^2}\sum_{j=1}^{B_i}\operatorname{Var}\left(I\left(\eta_i^j \leq t_i\right)\nabla f_i\left(x;\xi_i^j\right)\right),$$

then we use the fact that

$$\operatorname{Var}\left(XY\right) = \operatorname{Var}\left(X\right)\operatorname{Var}\left(Y\right) + \operatorname{Var}\left(X\right)\mathbb{E}\left[Y\right]^2 + \operatorname{Var}\left(Y\right)\mathbb{E}\left[X\right]^2,$$

where X and Y are independent random variables. Hence, we obtain the following bound on the variance

$$\begin{aligned}\operatorname{Var}\left(I\left(\eta_i^j \leq t_i\right)\nabla f_i\left(x;\xi_i^j\right)\right) &\leq p_i(1-p_i)\sigma^2 + p_i\left(1-p_i\right)\left\|\nabla f_i(x)\right\|^2 + \sigma^2 p_i^2 \\ &= p_i\sigma^2 + p_i\left(1-p_i\right)\left\|\nabla f_i(x)\right\|^2.\end{aligned}$$

As a result, the variance of $g(x)$ is bounded by

$$\begin{aligned}\operatorname{Var}\left(g(x)\right) &\leq \frac{1}{n^2}\sum_{i=1}^{n}\frac{1}{p_i^2 B_i^2}\sum_{j=1}^{B_i}\left(p_i\sigma^2 + p_i\left(1-p_i\right)\left\|\nabla f_i(x)\right\|^2\right) \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\frac{1}{p_i B_i}\left(\sigma^2 + \left(1-p_i\right)\left\|\nabla f_i(x)\right\|^2\right).\end{aligned}$$

Finally

$$\begin{aligned}\mathbb{E}\left[\left\|g(x)^2\right\|\right] &= \operatorname{Var}\left(g(x)\right) + \left\|\mathbb{E}\left[g(x)\right]\right\|^2 \\ &\leq \left\|\nabla f(x)\right\|^2 + \frac{1}{n^2}\sum_{i=1}^{n}\frac{1-p_i}{p_i B_i}\left\|\nabla f_i(x)\right\|^2 + \frac{\sigma^2}{n^2}\sum_{i=1}^{n}\frac{1}{p_i B_i}.\end{aligned}$$

Next we use $\left\|\nabla f_i(x)\right\|^2 \leq 2L\Delta$, thus

$$\mathbb{E}\left[\left\|g(x)^2\right\|\right] \leq \frac{2L\Delta}{n^2}\sum_{i=1}^{n}\frac{1-p_i}{p_i B_i} + \left\|\nabla f(x)\right\|^2 + \frac{\sigma^2}{n^2}\sum_{i=1}^{n}\frac{1}{p_i B_i}.$$

■

H.2. Proof of Theorem 9

The following lemma gives time complexity for any choice of $B_1, \ldots, B_n$ and $t = (t_1, \ldots, t_n)$ in Vecna SGD.

Lemma 17 (Proof in ?? H.2.1) *Assume that Assumptions 1, 2 hold for the function f and Assumption 3 holds for the function f_i for all $i \in [n]$. Let $\gamma = \min\left\{\frac{1}{\sqrt{LAK}}, \frac{1}{LB}, \frac{\varepsilon}{2LC}\right\}$ in Method 3. Then after*

$$T_{\text{VecnaSGD}}(t) \geq \max_{i\in[n]}\left\{B_i\left(\tau_i + t_i\right)\right\}\frac{12\Delta L}{\varepsilon}\max\left\{1, \frac{12\Delta A}{\varepsilon}, \frac{2C}{\varepsilon}\right\}$$

seconds, the method guarantees to find an ϵ-stationary point, where

$$A = \frac{L}{n^2}\sum_{i=1}^{n}\frac{1-p_i}{p_i B_i}, \quad C = \frac{\sigma^2}{n^2}\sum_{i=1}^{n}\frac{1}{p_i B_i}.$$

Now we are ready to prove the theorem.

Theorem 9 *Assume that Assumptions 1, 2 hold for the function f and Assumption 3 holds for the function f_i for all $i \in [n]$. Let $\gamma = \min\left\{\frac{1}{\sqrt{LAK}}, \frac{1}{LB}, \frac{\varepsilon}{2LC}\right\}$ in Method 3. Let $t = (t_1, \ldots, t_n)$, $t_1, \ldots, t_n \geq 0$. Without loss of generality assume that $0 < \tau_1 + t_1 \leq \cdots \leq \tau_n + t_n$. Let*

$$T = \tau_n + t_n + \left[\frac{1}{n}\sum_{i=1}^{n}\frac{\tau_i + t_i}{p_i}\right]\frac{\sigma^2}{n\varepsilon} + \left[\frac{1}{n}\sum_{i=1}^{n}\frac{1-p_i}{p_i}(\tau_i + t_i)\right]\frac{\Delta L}{n\varepsilon}.$$

Put

$$B_i = \lceil b_i \rceil, \quad b_i = \frac{T}{\tau_i + t_i}.$$

Then, Vecna SGD guarantees to find an ϵ-stationary point after

$$T_{\text{VecnaSGD}}(t) \geq 288 \times \frac{\Delta L}{\varepsilon}\left(\tau_n + t_n + \left[\frac{1}{n}\sum_{i=1}^{n}\frac{\tau_i + t_i}{p_i}\right]\frac{\sigma^2}{n\varepsilon} + \left[\frac{1}{n}\sum_{i=1}^{n}\frac{1-p_i}{p_i}(\tau_i + t_i)\right]\frac{\Delta L}{n\varepsilon}\right)$$

seconds.

Proof Since we have $b_i \geq 1$ for all $i \in [n]$, we get

$$\max_{i\in[n]}\{B_i(\tau_i + t_i)\} \leq \max_{i\in[n]}\{(b_i+1)(\tau_i + t_i)\} \leq 2\max_{i\in[n]}\{b_i(\tau_i + t_i)\} = 2T.$$

It remains to apply Theorem 17. We get

$$\begin{aligned}\frac{12\Delta A}{\varepsilon} &= \frac{12\Delta L}{\varepsilon n^2}\sum_{i=1}^{n}\frac{1-p_i}{p_i B_i} \leq \frac{12\Delta L}{\varepsilon n^2}\sum_{i=1}^{n}\frac{1-p_i}{p_i b_i}\\ &= \frac{12\Delta L}{n\varepsilon}\frac{1}{T}\frac{1}{n}\sum_{i=1}^{n}\frac{1-p_i}{p_i}(\tau_i + \eta_i) \leq 12,\end{aligned}$$

and

$$\frac{2C}{\varepsilon} = \frac{2\sigma^2}{\varepsilon n^2}\sum_{i=1}^{n}\frac{1}{p_i B_i} \leq \frac{2\sigma^2}{\varepsilon n^2}\sum_{i=1}^{n}\frac{1}{p_i b_i} \leq \frac{2\sigma^2}{n\varepsilon}\frac{1}{T}\frac{1}{n}\sum_{i=1}^{n}\frac{\tau_i + t_i}{p_i} \leq 2.$$

Finally, we get that Algorithm 3 returns a solution after

$$\begin{aligned}T_{\text{MindFlayerSGD}}(t) &\geq \max_{i\in[n]}\{B_i(\tau_i + t_i)\}\frac{12\Delta L}{\varepsilon}\max\left\{1, \frac{12\Delta A}{\varepsilon}, \frac{2C}{\varepsilon}\right\}\\ &\geq 288\frac{\Delta L}{\varepsilon}T\\ &\geq 288\frac{\Delta L}{\varepsilon}\left(\tau_n + t_n + \left[\frac{1}{n}\sum_{i=1}^{n}\frac{\tau_i + t_i}{p_i}\right]\frac{\sigma^2}{n\varepsilon} + \left[\frac{1}{n}\sum_{i=1}^{n}\frac{1-p_i}{p_i}(\tau_i + t_i)\right]\frac{\Delta L}{n\varepsilon}\right)\end{aligned}$$

seconds. ■

H.2.1. PROOF OF THEOREM 17

Lemma 17 *Assume that Assumptions 1, 2 hold for the function f and Assumption 3 holds for the function f_i for all $i \in [n]$. Let $\gamma = \min\left\{\frac{1}{\sqrt{LAK}}, \frac{1}{LB}, \frac{\varepsilon}{2LC}\right\}$ in Method 3. Then after*

$$T_{\text{VecnaSGD}}(t) \geq \max_{i\in[n]}\{B_i(\tau_i + t_i)\}\frac{12\Delta L}{\varepsilon}\max\left\{1, \frac{12\Delta A}{\varepsilon}, \frac{2C}{\varepsilon}\right\}$$

seconds, the method guarantees to find an ϵ-stationary point, where

$$A = \frac{L}{n^2}\sum_{i=1}^{n}\frac{1-p_i}{p_i B_i}, \quad C = \frac{\sigma^2}{n^2}\sum_{i=1}^{n}\frac{1}{p_i B_i}.$$

Proof Let $T_i^j(t_i)$ be the random time taken by client i in the j-th attempt of calculating gradient estimator. We have

$$T_i^j(t_i) = \begin{cases} \tau_i + \eta_i^j, & \text{if } \eta_i^j \leq t_i, \\ \tau_i + t_i, & \text{if } \eta_i^j > t_i. \end{cases} \tag{12}$$

Thus, the random time taken for client i to finish it's all B_i trials is

$$\mathcal{T}_i(t_i) := \sum_{j=1}^{B_i} T_i^j(t_i) \leq B_i(\tau_i + t_i). \tag{13}$$

Finally, let $\mathcal{T}$ be the random time required for one iteration of MindFlayer SGD. We get

$$\mathcal{T} = \max_{i\in[n]}\mathcal{T}_i(t_i) \leq \max_{i\in[n]}\{B_i(\tau_i + t_i)\}. \tag{14}$$

It remains to multiply $\mathcal{T}$ with the number of iterations K given by Theorem 8. ■

Appendix I. Experiments

In this section we explain the setup for comparing MindFlayer SGD, Rennala SGD, and ASGD, which we used throughout this paper. We compare the algorithms' performance on a quadratic optimization (15) task with access to a stochastic gradient. The parallelism was simulated on a machine with 2 Intel(R) Xeon(R) Gold 6226R CPUs @ 2.90GHz, with a total of 64 logical CPUs. For each setting of the algorithm, we run 10 different seeds for the random time and plot the average, minimum and maximum, see Figure 1, Figure 2, etc.

We use a similar setup to the one employed by Tyurin and Richtárik [24], but modify it so that we have a known expected variance. We make this choice, so we can compare theoretical parameters, as we did in Figure 2.

Furthermore, we consider the homogeneous optimization problem 1, with the convex quadratic function:

$$f(x) = \frac{1}{2}x^\top Ax - b^\top x \qquad \forall x \in \mathbb{R}^d.$$

We take $d = 1000$,

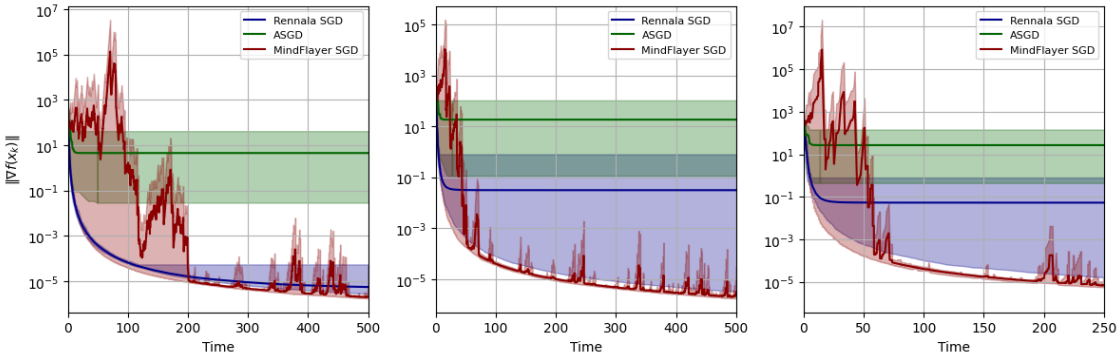

Figure 3: We ran an empirical experiment as descibed in the experiments section where we employ the same $\mathcal{J}_i = \text{InfBernoulli}(q)$ distribution for all clients $i \in [n]$, with different q values. From left to right we have $q = 0.6, 0.7, 0.8$. Additionally, we set $\tau_i = \sqrt{i+1}$. As we observe, with an increase of the probability of failure q unlike Rennala SGD and ASGD, MindFlayer SGD demonstrates the ability to continue optimizing and not be stuck

$$A = \frac{1}{4}\begin{bmatrix} 2 & -1 & 0 & \\ -1 & \ddots & \ddots & \ddots \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{d\times d} \quad \text{and} \quad b = \frac{1}{4}\begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{d}. \tag{15}$$

Assume that all n workers has access to the following unbiased stochastic gradients:

$$[\nabla f(x,\xi)]_j := \nabla_j f(x) + \xi,$$

where $\xi \sim \mathcal{N}(0, 0.0003^2)$, thus, we get that in Assumption 3 we have,

$$\sigma^2 = 0.0003^2 \cdot d = 0.0003^2 \cdot 1000.$$

Now setting the convergence threshold $\epsilon = 10^{-4}$, we can infer all theoretical parameters. To find the optimal time corresponding to Rennala SGD we need to fix the times, we do that by either removing the randomness, or adding the expected randomness.

On the other hand, for MindFlayer SGD we use the results from Theorem 5 to set the theoretical number of trials for each client.

In addition to the experimental results shown throughout the paper, we ran two more experiments. One with the Infinite-Bernoulli distribution on the same quadratic problem, and a second with the Log-Cauchy distribution with a small two-layer neural network on the MNSIT dataset [18].

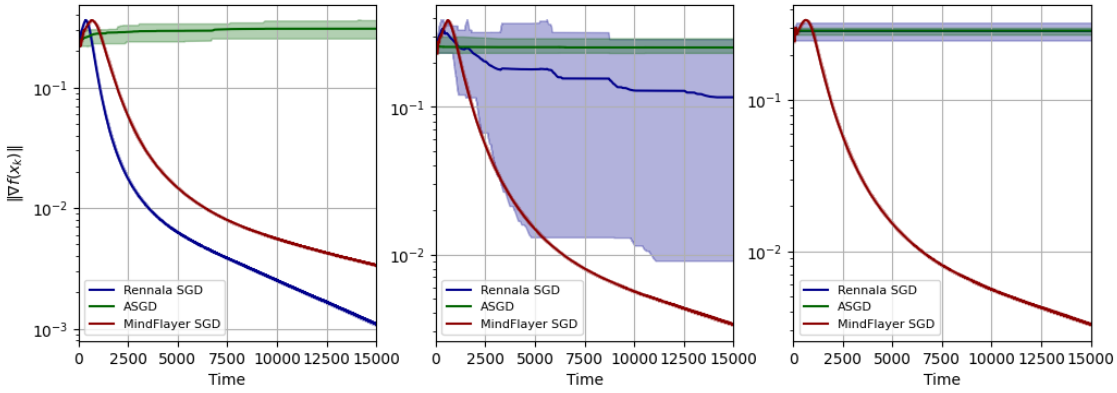

Figure 4: We train a two layer Neural Network on the MNIST dataset where we set the distribution $\mathcal{J}_i = \text{Log-Cauchy}(s)$ for all clients $i \in [n]$, with different scale values s. From left to right we have $s = 1, 10, 100$. Additionally, we set $\tau_i = \sqrt{i+1}$. We observe that Mindflayer SGD convergence doesn't suffer from the increase in the scale parameter s. On the other hand, Rennala and ASGD are delayed significantly with bigger scale parameters s