The Entrapment Problem in Random Walk Decentralized Learning

Zonghong Liu and Salim El Rouayheb
Department of Electrical and Computer Engineering
Rutgers University, New Brunswick, NJ, USA
Email: {zonghong.liu, salim.elrouayheb}@rutgers.edu

Matthew Dwyer
Network, Cyber, and Computational Sciences Division
DEVCOM Army Research Laboratory, Adelphi, MD, USA
Email: matthew.r.dwyer7.civ@army.mil

2024 IEEE International Symposium on Information Theory (ISIT) | 979-8-3503-8284-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISIT57864.2024.10619692

***Abstract*—This paper explores decentralized learning in a graph-based setting, where data is distributed across nodes. We investigate a decentralized SGD algorithm that utilizes a random walk to update a global model based on local data. Our focus is on designing the transition probability matrix to speed up convergence. While importance sampling can enhance centralized learning, its decentralized counterpart, using the Metropolis-Hastings (MH) algorithm, can lead to the entrapment problem, where the random walk becomes stuck at certain nodes, slowing convergence. To address this, we propose the Metropolis-Hastings with Lévy Jumps (MHLJ) algorithm, which incorporates random perturbations (jumps) to overcome entrapment. We theoretically establish the convergence rate and error gap of MHLJ and validate our findings through numerical experiments.**

I. Introduction

Traditional machine learning typically stores and trains models on a single server. This framework struggles with large-scale data and poses privacy leakage issues. These challenges have led to a shift towards researching distributed learning [1], [2]. A particularly focused framework is centralized distributed learning, which requires a central server, suffers from a communication bottleneck [3], and is vulnerable if the central server fails [4], [5]. Decentralized learning models remove the dependency on a central server. In this paper, we study the setting of decentralized learning via random walks (RWs), as shown in Fig. 1. The data needed to train the global model is held by local devices (nodes) in a network. Moreover, no central server is needed to aggregate the local updates performed in each iteration at the nodes [6], alleviating the problems of communication bottleneck, privacy, and failures that come with a centralized setting. The learning task is accomplished by leveraging the local communication links among the devices. The learning task can be expressed as:

$$\min_{x\in\mathbb{R}^d} \frac{1}{|V|}\sum_{v\in V} f_v(x), \tag{1}$$

where f_v is the local loss function of v, which depends on the local data. Existing decentralized learning approaches can be categorized into two main categories: gossip algorithms [7], [8], which have been extensively studied, and random walk algorithms [9], [10], which have been garnering increasing

This work was supported in part by the Army Research Lab (ARL) under Grant W911NF-21-2-0272 and the National Science Foundation (NSF) under Grant CNS-2148182.

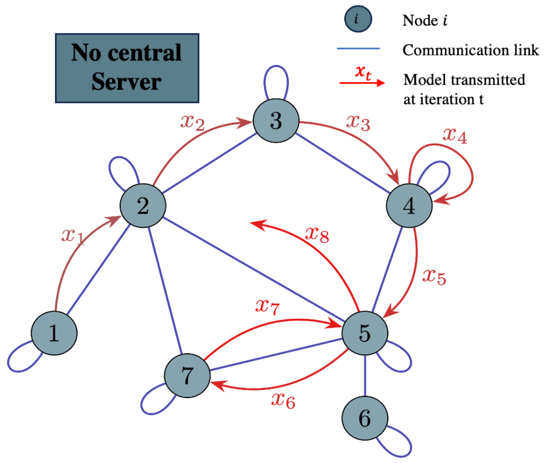

Fig. 1: Decentralized learning via random walk. The model x is carried by a random walk, which is represented by the red arrows. The model is updated using local data of the visited node in each iteration.

interest recently. In this work, we focus on random walk algorithms due to their overall low communication overhead. We study solving Eq. (1) using a random walk SGD method [11]–[13], that is, with the initial model x^0, starting node v_0, repeat the following steps: at iteration t, node v_t updates the model using the stochastic gradient $\hat{g}_{v_t}$ calculated based on the local data, and then passes the model to one of its neighbors randomly chosen according to a transition matrix P.

We focus on the effect of designing P on the algorithm's convergence. Three choices of P have been studied.

1) The simplest approach is to choose the next node u is chosen uniformly at random: $P(v,u) = \frac{1}{\deg(v)}$. However, the stationary distribution of this RW is proportional to the nodes' degrees.
2) The most extensively investigated approach is to employ the Metropolis-Hastings (MH) algorithm [14], [15] to construct P with the goal of achieving a uniform stationary distribution [16]: $P(v,u) = \frac{1}{\deg(v)}\min\{1, \frac{\deg(v)}{\deg(u)}\}, u \neq v, (u,v) \in E$. This design tries to mimic the vanilla centralized SGD that samples the data uniformly.
3) For a general desired stationary distribution π on the nodes, use the following transition probability obtained

from MH [10]:

$$P(v,u) = \min\left\{\frac{1}{\deg(v)}, \frac{\pi_u}{\deg(u)\pi_v}\right\}, u \neq v, (u,v) \in E.$$

We are interested in the last option, where the desired distribution π is set to be the importance sampling distribution. In centralized learning, sampling the data according to their "importance" may speed up the convergence [17]. We show in this work that using the Metropolis-Hastings algorithm to implement importance sampling in decentralized learning may cause a phenomenon we term the entrapment problem. Namely, the random walk may become entrapped in specific nodes or regions of the graph for an extended duration, thereby slowing down the convergence rate. We propose a new design strategy based on perturbing the Metropolis-Hastings transition probability P with Lévy-like jumps [18] to overcome entrapment and show that it can speed up the convergence.

A. Previous Work

The original work by [9], [16], [19] marked the initial exploration of random walk learning with the (sub-)gradient method. Subsequently, the work of [20] extended the results to accommodate changing topology networks. Utilization of curvature information to accelerate the convergence rate of random walk SGD was studied in [21]. Meanwhile, random walk SGD with adaptive step sizes was investigated in [22]. Ergodic sampling for the mirror descent method was explored in [23]. Under Markovian sampling, non-convex results for SGD were presented in [24], the AdaGrad method was examined in [25], variance reduction methods applicable to non-convex cases are investigated by [26].

Another direction relevant to this work is importance sampling. Needell et al. showed in [17] that sampling the data proportional to the gradient Lipschitz constant of the local loss function can speed up the convergence rate when the data is heterogeneous. The work in [27] connected importance sampling with minimizing the variance of the gradient estimator. Importance sampling for minibatches was studied in [28]. All these works focused on centralized scenarios and did not address the decentralized case we focus on here.

All the aforementioned work on random walk learning primarily studied scenarios where the random walk's stationary distribution is uniform across the nodes. The work in [10], [29] went beyond uniform sampling using importance sampling and multi-armed bandits.

B. Contributions

We investigate the impact of designing the transition probability of the random walk on the convergence properties of the random walk learning algorithm. We start by implementing importance sampling in a decentralized learning framework via the Metropolis-Hastings algorithm. Subsequently, we show that when the data is heterogeneous, and the network is not well-connected, the random walk governed by Metropolis-Hastings transition may become entrapped at certain "important" nodes. We call this phenomenon the entrapment problem. This entrapment phenomenon will force the model to be biased toward the local data and marginalize the updates. To mitigate this issue, we propose a novel algorithm, the Metropolis-Hastings with Lévy Jump (MHLJ), which incorporates random perturbations to help the random walk avoid getting entrapped. We then analyze the convergence rate and error gap of the MHLJ algorithm, supplementing our theoretical findings with simulations to validate our results.

C. Organization

The rest of the paper is organized as follows: Section II introduces the problem setting. Sections III and IV introduce the decentralized way of implementing importance sampling and the entrapment problem. Our proposed algorithm, MHLJ, to overcome the entrapment problem and the simulation results are introduced in Section V. Finally, we give our theoretical convergence result of MHLJ in Section VI. The complete proof of the theoretical convergence result and the simulation settings can be found in the Appendix of an extended version of this paper[1].

II. Problem Setting

A. Network and Objective Function

We consider a communication network represented by a connected graph $G = (V, E)$, where V is the set of nodes, and $E \subseteq V \times V$ represents the communication links between nodes. Nodes that are connected can communicate with each other. We assume that each node in the graph has a self-loop. Each node v of the network has its local data $x \in R^d$, which induces a local loss function $f_v(x)$. The goal is to find a decentralized algorithm to solve Eq. (1) using only local communications without the help of a central server. The objective function to minimize can be expressed as follows:

$$f(x) = \frac{1}{|V|}\sum_{v\in V} f_v(x),\ x \in \mathbb{R}^d. \tag{2}$$

B. Data Heterogeneity

We will work under the Lipschitz smooth assumption:

Definition 1. *A function $f(x)$ is L-smooth if*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|, \textit{ for all } x, y \in \textit{dom}(f),$$

where L is the gradient Lipschitz constant of the function.

For example, in linear regression $f_v(x) = \frac{1}{2}\|y_v - x^T A_v\|^2$ one can set $L_v = \|A_v\|^2$, and in Logistic regression $f_v(x) = y_v x^T A_v - \log(1 + e^{x^T A_v})$ one can set L_v can be chosen as $\frac{1}{4}\|A_v\|^2$, where (A_v, y_v) is the local data stored at node v.

We are interested in the scenario where the data owned by the nodes is heterogeneous, i.e., not sampled from identical distributions. We will look at the gradient Lipschitz constants L_v of the local loss functions f_v as a proxy for heterogeneity. We denote $L_{\max} = \max\{L_v | v \in V\}$, $L_{\min} =$

[1]https://github.com/ZonghongLiu/ISIT2024-Entrapment-Extended

$\min\{L_v | v \in V\}$, and $\bar{L} = \frac{1}{|v|}\sum_{v\in V} L_v$. In particular, we consider the following heterogeneous scheme:

$$L_{\min} \approx \bar{L} \leq L_{\max}. \tag{3}$$

In this case, importance sampling consisting of sampling data proportional to the local gradient Lipschitz constant can lead to a speed-up in convergence [17] .

C. Random Walk Learning

We want to design a decentralized algorithm that solves Eq. (1) via a random walk. A random walk algorithm for decentralized optimization [10] with a given transition probability matrix P consists of the following steps:

1) Start from a randomly selected node v_0, with the currently visited initial model x^0;
2) At iteration t, v_t updates the model using the stochastic gradient $\hat{g}_{v_t}$ calculated based on the local data:
$$x^{t+1} = x^t - \gamma_t \hat{g}_{v_t}(x^t), \tag{4}$$
3) Node v_t randomly chooses one of its neighbors (including itself) as v_{t+1}, according to a distribution $P(v_t, \cdot)$.
4) Node v_t passes the model x^{t+1} to node v_{t+1}.

The algorithm runs steps 2), 3), and 4) iteratively for a given number of iterations T. The model passed among the nodes and their neighbors can be seen as a time-homogeneous random walk on the graph G with transition matrix P. We assume that P is aperiodic and recurrent. Therefore, the random walk is ergodic and converges to a stationary distribution π.

III. Importance Sampling

Our main objective is to design the transition matrix P of the random walk to speed up the convergence of the learning algorithm. Our approach is to mimic centralized importance sampling, which has been shown to improve convergence in certain regimes [17], [27]. The main challenge is that the data sampled by the random walk is not i.i.d anymore, but is governed by a Markovian dependency imposed by the graph.

A. Importance Sampling in Centralized Learning

Importance sampling has been mainly studied in the centralized setting. In the standard SGD algorithm, the data is sampled uniformly. Importance sampling goes beyond the uniform distribution and samples the data based on a certain measure of importance, still in i.i.d. fashion. Of particular importance to our work here is the work of Needell et al. [17], where it was shown that weighted sampling in a centralized setting can speed up the convergence of SGD. It was proposed to use the gradient Lipschitz constant of the local loss function L_i as the importance of data x_i, and to sample the data proportional to its importance, i.e., according to the following distribution:

$$\pi_I(i) := \frac{L_i}{\sum_{i=1}^{N} L_i}, \tag{5}$$

where π_I is defined to be the importance sampling distribution. The following convergence rates of SGD for Lipschitz smooth and strongly convex objective functions were shown in [17, Theorem 2.1]:

1) Uniform Sampling: $\tilde{\mathcal{O}}(\frac{L_{\max}}{T})$;
2) Importance Sampling: $\tilde{\mathcal{O}}(\frac{\bar{L}^2}{L_{\min}T})$,

where $L_{\max}$, $L_{\min}$, and $\bar{L}$ are the maximum, minimum, and average of local gradient Lipschitz constants, respectively. From the convergence rate, we see that when Eq. (3) holds, i.e., when the gap between $L_{\max}$ and $\bar{L}$ is significant, and $L_{\min}$ is close to $\bar{L}$, sampling according to the importance distribution in Eq. (5) will speed up the convergence of SGD.

B. Importance Sampling in Decentralized Learning

In a graph-based decentralized setting, there is no central server to implement importance sampling. Ayache et al. [10] proposed importance sampling in random walk learning by designing the transition probability P to achieve a desired stationary distribution $\pi = \pi_I$, which is proportional to L_i as in Eq. (5), via Metropolis-Hastings algorithm. Given a distribution π, the MH algorithm allows designing a transition matrix P that has π as its stationary distribution:

$$P(i,j) = \begin{cases} Q(i,j)\min\{1, \frac{\pi(j)Q(j,i)}{\pi(i)Q(i,j)}\}, & i \neq j, \\ 1 - \sum_{k:(i,k)\in E} P(i,k), & i = j, \end{cases} \tag{6}$$

where Q is any proper transition probability that satisfies the graph structure, i.e., $Q(i,j) = 0$ if $(i,j) \notin E$, $Q^k(i,j) > 0$ for some k if there is a path from i to j. For example, we can take Q as the simple random walk on the graph, i.e., the neighbors are selected uniformly $Q(i,j) = \frac{1}{\deg(i)}, \forall (i,j) \in E$.

To mimic importance sampling, i.e., $\pi(i) \propto L_i$, the transition probability matrix can be chosen to be [10]:

$$P_I(i,j) = \begin{cases} \frac{1}{\deg(i)}\min\{1, \frac{\deg(i)L_j}{\deg(j)L_i}\}, & i \neq j, \\ 1 - \sum_{k:(i,k)\in E} P(i,k), & i = j. \end{cases} \tag{7}$$

IV. The Entrapment Problem

In certain cases, the Metropolis-Hastings importance sampling transition given by Eq. (7) can lead to a degradation in the convergence rate. We show that when the data is heterogeneous, and the graph is not "well-connected", the random walk moving according to Eq. (7) may get entrapped in a local area of the graph, leading to a slowdown in convergence.

We will illustrate our ideas using the example of a ring network with heterogeneous data. Fig. 2.a gives such an

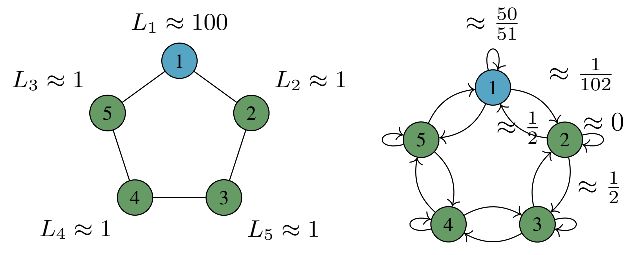

Fig. 2: (a) An example of ring topology with five nodes that may cause the entrapment issue. (b) In the Markov chain representation of the random walk on the graph in (a).

example where the data are stored over a ring network with 5 nodes. Here, node 1 stores the data set that has a much larger gradient Lipschitz constant. We show that in this case, the convergence rate of Metropolis-Hastings importance sampling is dramatically slowed down, as shown in Fig. 3. The reason is that the random walk is getting entrapped on the "important" nodes, i.e., nodes holding data with large L_i's. This forces the algorithm to update the model using the same data a large number of times, pushing the model to converge to the local optimum, thus slowing down convergence.

To understand the cause of the entrapment problem, notice that the transition probability P_I of the random walk given in Eq. (7) satisfies the detailed balanced condition $\pi(i)P_I(i,j) = \pi(j)P_I(j,i)$ [30], which in our case leads to

$$L_i/L_j = P_I(j,i)/P_I(i,j). \tag{8}$$

Therefore, when a node has much larger local gradient Lipschitz constants than its neighbors', and the graph is sparse, the probability of leaving this node is very small.

We observe that the entrapment problem does not occur only in the ring network but also in other "sparse" networks, like $2D$-grids and Watts-Strogatz random graphs.

V. MHLJ Algorithm

We propose a new algorithm, Metropolis-Hastings with Lévy Jumps (MHLJ), to solve the entrapment problem. The main idea consists of perturbing the Metropolis-Hastings transition probability in Eq. (7) by adding random jumps to escape a local entrapment. The added jump requires no global information on the graph. Each step of the jump requires only local structure information, i.e., the neighbors of the current node. The details are described in Algorithm 1, where (p_J, p_d, r) are the parameters of the Lévy jumps, and $\mathcal{N}_v$ is the neighbor set of node v. We compare the performance of Uniform sampling via MH, Importance sampling via MH, and Importance sampling via MHLJ in Fig. 3 for the ring network with 1000 nodes. The simulation results show the following:

i. MHLJ can break the entrapment and significantly speed up the convergence rate.
ii. MHLJ exhibits asymptotically an error gap that we will later explain in our theoretical analysis.

Algorithm 1 Importance Sampling using Metropolis-Hastings with Lévy Jumps (MHLJ)

```
Input: $G = (V, E)$, $L_v$ for $v \in V$, $P_I$, $\gamma$, $T$, $p_J$, $p_d$, $r$
Output: $x^T$
    Initialisation: $x^0$, $v_0$,
 1: for $t = 0, 1, T-1$ do
 2:     $x^{t+1} = x^t - \gamma \frac{\bar{L}}{L_{v_t}} \nabla f_{v_t}(x^t)$
 3:     $J \sim Ber(p_J)$
 4:     if $J = 0$ then
 5:         $v_{t+1} \sim P_I(v_t, \cdot)$
 6:     else
 7:         $d \sim \mathsf{TruncGeom}(p_d, r)$
 8:         while $d \geq 0$ do
 9:             $v_{t+1} \sim Unif(\mathcal{N}_{v_t})$
                $v_t = v_{t+1}$
                $d = d - 1$
10:         end while
11:     end if
12: end for
13: return $x^T$
```

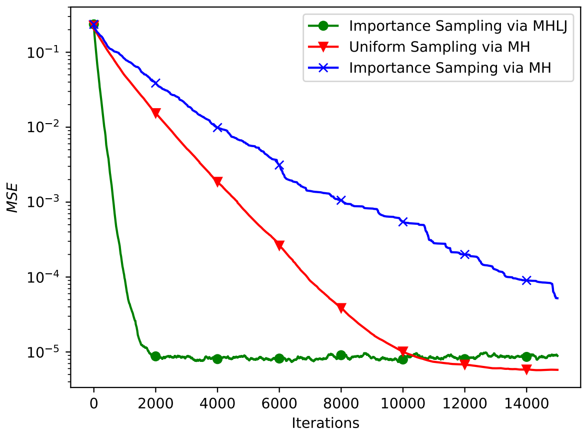

Fig. 3: Linear regression model $y = Ax + \epsilon$ trained on a synthetic heterogeneous data set over a ring network with 1000 nodes. We compare the uniform sampling, importance sampling, and our Algorithm MHLJ. The y-axis is the mean square error (MSE), i.e., $\sum_{v \in V} \|y_v - A_v \hat{x}\|^2/|V|$. The x-axis is the number of iterations with SGD updates, i.e., the number of times Eq. (4) is called. We generate the data A_v on node v with $A_v \overset{\text{i.i.d.}}{\sim} N(0, \sigma^2 \mathbb{I}_{10})$, where σ^2 takes value 1 with probability $p = 0.998$ and 100 with probability $p = 0.002$. The noise is generated from $\epsilon \overset{\text{i.i.d.}}{\sim} N(0,1)$. We use the hyper-parameters: $(p_J, p_d, r) = (0.1, 0.5, 3)$.

In MHLJ the random walk determines its next step after each update. Specifically, it either executes a Lévy jump with a probability of p_J or adheres to the Metropolis-Hastings rule with a probability of $1 - p_J$.

Lévy jump: When the random walk makes a jump: (a) The random walk chooses how far it should jump. The jumping distance d is sampled from a truncated Geometric (TrunGeom) distribution defined by $P(D = d) = \frac{p_d(1-p_d)^{d-1}}{1-(1-p_d)^r} \mathbb{I}\{d \leq r\}$. (b) Once the distance d is determined, the model undergoes d consecutive transfers between nodes, wherein it is passed to a uniformly selected neighboring node d times in succession without undergoing any updates. The simple random walk strategy employed during the jumps is deliberately designed to disrupt the detailed balance condition, thereby enabling the random walk to escape the entrapping region. As a consequence, the sampling distribution of nodes deviates from the desired importance distribution defined in Eq. (5), resulting in an error gap, which will also appear in our convergence result presented in Theorem 1.

Algorithm 1 induces a time-homogeneous random walk with transition matrix P. We view this random walk as a Metropolis-Hastings random walk (with transition matrix P_I

defined in Eq. (7)) perturbed by Lévy jumps, i.e.,

$$P = (1 - p_J)P_I + p_J P_{\textit{Lévy}},\ \textit{where}$$
$$P_{\textit{Lévy}} = \sum_{i=1}^{r} \frac{p_d(1-p_d)^{i-1}}{1-(1-p_d)^r}\, \mathrm{diag}\{A_G^i \mathbf{1}\}^{-1} A_G^i,$$

where A_G is the adjacency matrix of the given graph G. The resulting stationary distribution π is thus no longer $\pi_I(v) = \frac{L_v}{\sum_{v\in V} L_v}$ but a perturbed version of it.

Remark 1 (Computation v.s. Communication overheads of MHLJ). *Each iteration in MHLJ (x-axis in Fig. 3) corresponds to one gradient decent update according to Eq.* (4). *Fig. 3 shows that MHLJ saves on computation cost since it requires less updates to achieve a given accuracy. However, by adding jumps, we actually admit transitions without updates, which leads to an increase in the communication overhead. For each update, the expected number of transitions (node visits) required can be bounded by*

$$(1-p_J)\cdot 1 + p_J \mathbb{E}[d] \le 1 + p_J(\frac{1}{p_d} - 1).$$

In our example, this upper bound is equal to 1.1*, i.e., at most* 10% *increase in the average communication cost in our example.*

VI. Convergence Result

Now, we give our theoretical convergence result.

Theorem 1 (Convergence of Algorithm MHLJ). *Suppose that each local loss function f_v is L_v-smooth and μ-strongly convex, and $\|\nabla f_v(x^*)\|^2 \le \sigma_*^2$, $\forall v \in V$, then for $\gamma < \min\{\frac{1}{\bar{L}}, \frac{1}{T\mu}\ln T \frac{\|x^0-x^*\|^2\mu^2}{\tau_{mix}\sigma_*^2 \bar{L}}\}$, the output of Algorithm 1 after T iterations x^T satisfies:*[2]

$$\mathbb{E}\|x^T - x^*\|^2 \le \tilde{\mathcal{O}}\left(\frac{\bar{L}^2\tau_{mix}\sigma_*^2}{L_{\min}T}\right) + \mathcal{O}\left(p_J^2\|P_I - P_{\textit{Lévy}}\|_1^2\right) \tag{9}$$

where τ_{mix} is the mixing time of $P = P_I - p_J(P_I - P_{\textit{Lévy}})$, and $\bar{L} = \sum_{v\in V} L_v/|V|$.

The first term in Eq. (9) implies that the algorithm converges with a sub-linear rate. Here, τ_{mix} is the mixing time [30] of the random walk and represents the effect of sampling dependency induced by the graph topology. Also, note that τ_{mix} is smaller than its Metropolis-Hastings counterpart because making jumps makes the graph better connected. The second term describes the error gap caused by the jumps. The choice of p_J creates a trade-off between the speed with which the random walk can escape from the entrapment and the magnitude of the error gap expressed in the second term of Eq. (9). When the value of p_J is small, the random walk experiences difficulty escaping the entrapment, resulting in a slow convergence; conversely, a large value of p_J yields a more substantial error gap. As for $\|P_I - P_{\textit{Lévy}}\|_1$, its value depends on the graph and the gradient Lipschitz constants, and can be upper bounded by n^2. In practice, the error gap can be made arbitrarily small by decreasing p_J as the number of iterations increases.

[2] $\tilde{\mathcal{O}}$ hides logarithmic factors.

The proof of Theorem 1 presents two challenges compared to the standard proof of SGD:

1) The stochastic gradient $\nabla f_{v_t}(x^t)$ used in each update step is not an unbiased estimator of the true gradient due to the graph topology, i.e., $\mathbb{E}[\nabla f_{v_t}(x^t) \mid v_{t-1}] \ne \nabla f(x^t)$. Thus, each step is not a descent step in expectation as in standard SGD.
2) The detailed balance equation is violated by the added Lévy jumps, causing the expectation with respect to the stationary distribution to be also biased, i.e., $\mathbb{E}_{\pi_I}[\nabla f_v(x^*)] \ne 0$. This breaks the first order optimality condition .

To address the first challenge, we use an auxiliary sequence $\{y^t\}_{t=1}^T$ to bound $\|x^T - x^*\|$ without relying on the conditional unbiasedness of the gradient estimate. This proof technique was first introduced in [31] to study the random reshuffling method, and then used for the proof of Markovian SGD in [26]. Namely, we construct $\{y^t\}_{t=1}^T$ by letting:

$$y^{t+1} = y^t - \gamma \frac{\bar{L}}{L_{v_t}} \nabla f_{v_t}(x^*). \tag{10}$$

The following lemma controls the distance between x^t and y^t and is adapted from Lemma 9 in [26] to incorporate the smoothness constants.

Lemma 1. *For any $\{y^t\}_{t=0}^T$ satisfies Eq.* (10)*, we have*

$$\|x^{t+1} - y^{t+1}\|^2 \le (1-\gamma\mu)\|x^t - y^t\|^2 + \gamma\bar{L}\|y^t - x^*\|^2.$$

By setting $y^T = x^*$, we can upper bound $\|x^t - x^*\|^2$ by

$$\mathbb{E}\left[\|x^T - x^*\|^2\right] \le 2(1-\gamma\mu)^T\|x^0 - x^*\|^2 + 3\gamma^3\bar{L}\sum_{t\le T}(1-\gamma\mu)^{T-t}\underbrace{\mathbb{E}\left[\left\|\sum_{t\le s\le T}\frac{L_{v_s}}{\bar{L}}\nabla f_{v_s}(x^*)\right\|^2\right]}_{\textit{Accumulated error term}}. \tag{11}$$

In the case of an MH random walk with no jumps, the accumulated error term in Eq. (11) should converges to zero as $T \to \infty$ due to ergodicity. To address the second challenge, we prove in Lemma 2 an upper bound on the accumulated error term for MHLJ.

Lemma 2. *For $1 \le s \le t \le T$, we have*

$$\mathbb{E}\left[\left\|\sum_{i=s}^{t}\frac{L_v}{\bar{L}}\nabla f_{v_i}(x^*)\right\|^2\right] \le (t-s)C\tau_{mix}\sigma_*^2 + 2(t-s)^2 p_J^2\|P_I - P_{\textit{Lévy}}\|_1^2\sigma_*^2\left(\frac{\bar{L}}{L_{\min}}\right)^2.$$

Lemmas 1 and 2 serve as essential blocks for completing the proof of Theorem 1.

References

[1] M. Zinkevich, M. Weimer, L. Li, and A. Smola, “Parallelized stochastic gradient descent,” *Advances in neural information processing systems*, vol. 23, 2010.

[2] P. Richtárik and M. Takáč, “Parallel coordinate descent methods for big data optimization,” *Mathematical Programming*, vol. 156, pp. 433–484, 2016.

[3] S. Praneeth Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. Theertha Suresh, “Scaffold: Stochastic controlled averaging for federated learning,” *arXiv e-prints*, pp. arXiv–1910, 2019.

[4] V. Gupta, A. Ghosh, M. Derezinski, R. Khanna, K. Ramchandran, and M. Mahoney, “Localnewton: Reducing communication bottleneck for distributed learning,” *arXiv preprint arXiv:2105.07320*, 2021.

[5] R. Guerraoui, S. Rouault *et al.*, “The hidden vulnerability of distributed learning in Byzantium,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[6] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, “Advances and open problems in federated learning,” *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Gossip algorithms: Design, analysis and applications,” in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3. IEEE, 2005, pp. 1653–1664.

[8] ——, “Randomized gossip algorithms,” *IEEE transactions on information theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[9] B. Johansson, M. Rabi, and M. Johansson, “A randomized incremental subgradient method for distributed optimization in networked systems,” *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1157–1170, 2010.

[10] G. Ayache and S. El Rouayheb, “Private weighted random walk stochastic gradient descent,” *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 452–463, 2021.

[11] H. Robbins and S. Monro, “A stochastic approximation method,” *The annals of mathematical statistics*, pp. 400–407, 1951.

[12] D. P. Bertsekas *et al.*, “Incremental gradient, subgradient, and proximal methods for convex optimization: A survey,” *Optimization for Machine Learning*, vol. 2010, no. 1-38, p. 3, 2011.

[13] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, “Robust stochastic approximation approach to stochastic programming,” *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[14] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of state calculations by fast computing machines,” *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[15] W. K. Hastings, “Monte Carlo sampling methods using Markov chains and their applications,” 1970.

[16] B. Johansson, M. Rabi, and M. Johansson, “A simple peer-to-peer algorithm for distributed optimization in sensor networks,” in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 4705–4710.

[17] D. Needell, R. Ward, and N. Srebro, “Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm,” *Advances in neural information processing systems*, vol. 27, 2014.

[18] A. P. Riascos and J. L. Mateos, “Long-range navigation on complex networks using Lévy random walks,” *Physical Review E*, vol. 86, no. 5, p. 056110, 2012.

[19] C. G. Lopes and A. H. Sayed, “Incremental adaptive strategies over distributed networks,” *IEEE transactions on signal processing*, vol. 55, no. 8, pp. 4064–4077, 2007.

[20] S. S. Ram, A. Nedić, and V. V. Veeravalli, “Incremental stochastic subgradient algorithms for convex optimization,” *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 691–717, 2009.

[21] H.-T. Wai, N. M. Freris, A. Nedic, and A. Scaglione, “Sucag: Stochastic unbiased curvature-aided gradient method for distributed optimization,” in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 1751–1756.

[22] T. Sun, D. Li, and B. Wang, “Adaptive random walk gradient descent for decentralized optimization,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 20 790–20 809.

[23] J. C. Duchi, A. Agarwal, M. Johansson, and M. I. Jordan, “Ergodic mirror descent,” *SIAM Journal on Optimization*, vol. 22, no. 4, pp. 1549–1578, 2012.

[24] T. Sun, Y. Sun, and W. Yin, “On Markov chain gradient descent,” *Advances in neural information processing systems*, vol. 31, 2018.

[25] R. Dorfman and K. Y. Levy, “Adapting to mixing time in stochastic optimization with Markovian data,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 5429–5446.

[26] M. Even, “Stochastic gradient descent under Markovian sampling schemes,” *arXiv preprint arXiv:2302.14428*, 2023.

[27] P. Zhao and T. Zhang, “Stochastic optimization with importance sampling for regularized loss minimization,” in *international conference on machine learning*. PMLR, 2015, pp. 1–9.

[28] D. Csiba and P. Richtárik, “Importance sampling for minibatches,” *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 962–982, 2018.

[29] G. Ayache, V. Dassari, and S. El Rouayheb, “Walk for learning: A random walk approach for federated learning from heterogeneous data,” *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 929–940, 2023.

[30] D. A. Levin and Y. Peres, *Markov chains and mixing times*. American Mathematical Soc., 2017, vol. 107.

[31] K. Mishchenko, A. Khaled, and P. Richtárik, “Random reshuffling: Simple analysis with vast improvements,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 309–17 320, 2020.

[32] X. Mao, K. Yuan, Y. Hu, Y. Gu, A. H. Sayed, and W. Yin, “Walkman: A communication-efficient random-walk algorithm for decentralized optimization,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 2513–2528, 2020.

[33] H. Hendrikx, “A principled framework for the design and analysis of token algorithms,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2023, pp. 470–489.

[34] T. Jiang, C. Sun, S. El Rouayheb, and D. Pompili, “Facegroup: Continual face authentication via partially homomorphic encryption & group testing,” in *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE, 2023, pp. 443–451.

[35] A. Naor, S. Rao, and O. Regev, “Concentration of Markov chains with bounded moments,” *Ann. Inst. H. Poincaré Probab. Statist.*, vol. 56, no. 3, 2020.

[36] E. Seneta, “Perturbation of the stationary distribution measured by ergodicity coefficients,” *Advances in Applied Probability*, vol. 20, no. 1, pp. 228–230, 1988.

[37] A. Nedić, A. Olshevsky, and M. G. Rabbat, “Network topology and communication-computation tradeoffs in decentralized optimization,” *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.