Communication-efficient Decentralized Deep Learning

Fateme Fotouhi
Department of Mechanical Engineering
Department of Computer Science
Iowa State University, Ames, IA
fotouhif@iastate.edu

Aditya Balu
Department of Mechanical Engineering
Iowa State University, Ames, IA
baditya@iastate.edu

Zhanhong Jiang
Johnson Controls International
507 East Michigan St, Milwaukee, WI
starkjiang@gmail.com

Yasaman Esfandiari
HRL Laboratories
3011 Malibu Canyon Rd, Malibu, CA
yesfandiari@hrl.com

Salman Jahani
SAP Labs, LLC
3366 Via Lido, Newport Beach, CA
jahani.ie@gmail.com

Soumik Sarkar
Department of Mechanical Engineering
Department of Computer Science
Iowa State University, Ames, IA
soumiks@iastate.edu

Abstract

Decentralized deep learning algorithms leverage peer-to-peer communication of model parameters and/or gradients over communication graphs among the learning agents with access to their private data sets. The majority of the studies in this area focuses on achieving high accuracy, many at the expense of increased communication overhead among the agents. However, large peer-to-peer communication overhead often becomes a practical challenge, especially in harsh environments such as for an underwater sensor network. In this paper, we aim to reduce the communication overhead while achieving similar performance as the state-of-the-art algorithms. To achieve this, we use the concept of Minimum Connected Dominating Set from graph theory that is applied in ad hoc wireless networks to address communication overhead issues. Specifically, we propose a new decentralized deep learning algorithm called minimum connected Dominating Set Model Aggregation (**DSMA**). We investigate the efficacy of our method for different communication graph topologies with a small to a large number of agents using varied neural network model architectures. Empirical results on benchmark data sets show a significant (up to 100X) reduction in communication time while preserving the accuracy or in some cases increasing it compared to the state-of-the-art methods. We also present analysis to show convergence of our proposed algorithm.

1 Introduction

Over the past several years, researchers have been working on proposing various algorithms for distributed deep learning, among which Federated Learning (*FL*) (McMahan et al., 2017; Kairouz et al., 2019) shows promising performance in various scenarios. However, a major limitation with using *FL* is its dependence on a central parameter server. To tackle this limitation, decentralized deep learning (DDL) emerged as a branch of distributed deep learning. In DDL, several spatially distributed agents, each with an individual subset of data collectively learn together a deep learning

2022 Trustworthy and Socially Responsible Machine Learning (TSRML 2022) co-located with NeurIPS 2022.

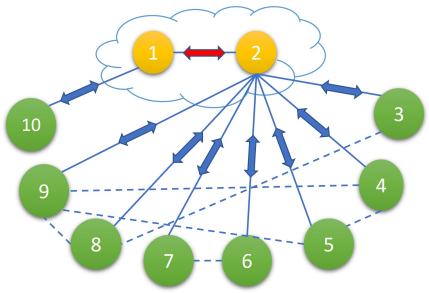


Figure 1: **DSMA** algorithm: Considering the graph topology network (solid and dotted blue lines) with 10 agents, The DSMA steps are as follows: (1) Finding the minimum connected dominating set of the network (yellow nodes). (2) Dominating agents (yellow nodes) communicate with dominated agents (green nodes) to send and receive model parameters. (3) Each agent computes weighted average of the received model parameters and its own parameters. (4) Dominating agents communicate with each other to create a consensus model. (5) All agents update their model parameters.

model by peer-to-peer communication of useful information such as the model parameters or the gradients. The major advantage of DDL is that, unlike *FL*, there is no centralized server and hence avoids any issues with server failure and server latency. While SoTA DDL algorithms work well for the smaller number of agents, there is significant communication overhead especially for larger number of agents such as $N > 50$. Different strategies for DDL have been proposed to reduce the communication overhead including compression (Lu and De Sa, 2020; Vogels et al., 2020), adding local updates in some iterations (Li et al., 2019), randomly choosing agents to communicate (Nadiradze et al., 2021), and changing communication graph topology (Wang et al., 2019). However, maintaining high accuracy and reducing the communication overhead simultaneously still remains a major challenge in the DDL literature. Additionally, these algorithms usually suffer from low accuracy for large-scale networks.

Communication reduction techniques have been widely investigated in Wireless Ad Hoc Network (WANET) and Mobile Ad Hoc Network (MANET) to find efficient network routing. One of the popular techniques involves finding Minimum Connected Dominating Set (MCDS) of the network graph topology and centralizing the whole network to a small set which is dominating the whole graph (Yu et al., 2013; Wang et al., 2021; Wu and Li, 1999). In the communication network literature, MCDS consists of nodes that are called gateway hosts. With this setup, every node is either a gateway host or connected to a host. Thus, only gateway hosts keep routing information in the network.

Inspired by the MCDS notion in Ad Hoc networks to reduce communication overhead, we propose a new algorithm for DDL called minimum connected Dominating Set Model Aggregation (DSMA). Our proposed algorithm not only targets communication reduction in DDL but also preserves the accuracy even in large-scale networks. At each training loop, each dominating agent communicates with its neighbors (that are not in the MCDS) to send and receive the model parameters. Then, each agent aggregates the received model parameters with its own model parameter using a weighted averaging scheme. After that, dominating nodes communicate with each other and average the model parameters to develop a consensus model among the MCDS agents (similar to other DDL methods) (Nadiradze et al., 2021). In the end, all agents update their model parameters. We illustrate the steps of our algorithm in Figure 1, where the green nodes, i.e., the dominated agents do not need to communicate with each other, thus reducing communication. At the same time, dominating agents can collect the whole graph model parameters to create a consensus model and then broadcast it to the dominated nodes in the next iteration. This ensures a complete cycle of transferring information among agents (for achieving consensus in the whole graph). This characteristic of MCDS (in our algorithm) is similar to that of a server in *FL* algorithms.

In summary, our main contributions in this paper are: (1) We propose a new graph-based communication protocol for decentralized learning systems to reduce the communication cost with minimal loss in performance. (2) We empirically verify our proposed algorithm for different datasets and different number of agents to demonstrate up to 100X reduction in communication cost compared to traditional decentralized learning methods. (3) Finally, we analyze how our communication protocol only transforms the optimization problem, but the spectral properties of the agent interaction remain similar.

Table 1: Comparisons between different decentralized deep learning approaches. Conv: convergence rate, Comm: Communication rate; Scalability: if the method can achieve high accuracy ($> \%50$) for $N > 50$; Comm.Red.Tech: Communication reduction technique; b: batch size; : number of non-zero elements in the graph connectivity matrix; ": cost regarding forward and backward pass during training the neural network; I_1: is the number of local updates; I_2: number of decentralized SGDs updates; : number of connections between agents that are not in connected dominating set S; z: probable missing connections between dominated and dominating sets.

Method	Conv	Comm	Scalability	Comm.Red.Tech
DPMSGD	$O(\frac{1}{T} + p\frac{1}{\sqrt{NT}})$	$O(b\rho + \varepsilon)$	✔	-
SGP	$O(\frac{1}{T} + p\frac{1}{\sqrt{NT}} + \frac{1}{T^{1.5}})$	$O(b\rho + \varepsilon)$	✖	approximate averaging
Matcha	$O(\frac{N}{T} + p\frac{1}{\sqrt{NT}})$	$O(\frac{b\rho}{2} + \varepsilon)$	✖	graph-based
LDSGD	$O(\frac{1}{T} + p\frac{1}{\sqrt{NT}})$	$O(\frac{I_2 b\rho}{I_1+I_2} + \varepsilon)$	✔	local updates
SwarmSGD	$O(\frac{1}{T})$	$O(\frac{b\rho}{2} + \varepsilon)$	✖	random sampling
DSMA	$O(\frac{1}{T} + p\frac{1}{\sqrt{NT}})$	$O(b(\rho \quad + z) + \varepsilon)$	✔	graph-based

1.1 Related Works

Large-scale deep learning relies heavily on parallel stochastic gradient descent (SGD). The majority of distributed deep learning methods i.e. *FL* take advantage of this method using a parameter server (Li et al., 2014) which results in a notable bandwidth cost that affects scalability. Also, in some applications, continuous communication between the agents and a central server is not feasible (Lian et al., 2017; Esfandiari et al., 2021). In an attempt to find a decentralized learning approach over a network of agents, several decentralized SGD approaches were proposed by researchers (Sun et al., 2021) that can be divided into different categories. One category is gossip averaging algorithms (Kempe et al., 2003; Lian et al., 2017) that try to achieve a partial average between the connected nodes. Among gossip averaging methods, DPMSGD (Lian et al., 2017) shows that their decentralized method can outperform centralized algorithms by avoiding the communication traffic jam. In another line of work, Scaman et al. (2018) introduced a multi-step primal-dual algorithm and analytically showed that the error decreases at a fast rate even for non-strongly-convex objective functions. Also, to accelerate the training process, Assran et al. (2019) proposed the SGP algorithm which converges at the same sub-linear rate as SGD and performs approximate distributed averaging. In the same line of work, Lu and De Sa (2020) and Vogels et al. (2020) proposed compression-based algorithms to improve memory usage and reduce the communication overhead of existing decentralized learning approaches. With the goal to address the data non-IIDness in decentralized learning, Esfandiari et al. (2021) proposed projecting the aggregated gradients from the neighboring agents into a single gradient using quadratic programming methods. SwarmSGD was also proposed by Nadiradze et al. (2021) which learns from random interactions between connected nodes in a graph to achieve consensus while reducing the communication overhead. Similarly, Tang et al. (2020) proposed an algorithm where each agent adaptively selects its peer based on the bandwidth resources. With the same goal, Li et al. (2019) proposed the LDSGD algorithm by considering a specific ratio of local to distributed iterations for training the agents' models. However, local updates result in a significant accuracy reduction in large-scale networks compared to the vanilla gossip-based averaging method.

As such, the communication overhead is significantly higher for DDL (unlike FL). Researchers have come up with algorithms that achieve the same linear speed up compared to centralized algorithms (Koloskova et al., 2020; Yu et al., 2019), and the training time is proportionate to the number of training agents (Ying et al., 2021).The communication overhead between the agents is heavily dependent on the graph topology, which determines what nodes can communicate with each other during training. The sparser the graph becomes, the less overall communication happens among the agents. Matcha (Wang et al., 2019) tries to use this concept and considers a subgraph (matching in the graph) to communicate in each iteration. However, using this method, consensus is achieved late and competitive accuracy even with a less number of agents is not guarenteed. Contrary to this approach, in this paper, we are looking more closely at the graph characteristics to propose the DSMA algorithm, which considers MCDS to dominate all agents and to help flow information in the graph. Therefore, this algorithm helps to eliminate excessive information transfer in the graph. Convergence and communication rates of benchmark methods are compared to our proposed algorithm in Table 1.

(a) (b) (c) (d)

Figure 2: Minimum connected dominating set (red) for different topologies (a) Fully Connected (FC), (b) Ring, (c) Bipartite, and (d) Random

2 Preliminaries

2.1 Decentralized Deep Learning

In DDL, N agents communicate to solve the following empirical risk minimization problem:

$$\min f(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{1}$$

where x 2 R^d is the model parameter, $f : \mathsf{R}^d$! R is the given loss function and $f_i(x)$ is the loss function corresponding to agent i which is defined based on its private dataset D_i. Although each agent works on its private training data concurrently, the goal for decentralized learning is to come up with a consensus model for the agents through a round of communication during the updating process. The communication between the agents happens through a connected undirected graph topology $G(V, E)$, where edges E are communication links, and V indicates vertex nodes or agents in this framework. Agent a 2 V and agent b 2 V are considered neighbors and can only communicate if (a, b) 2 E. Further, most DDL approaches also define communication matrix 2 $\mathsf{R}^{N\ N}$ representing the agent's interactions, where each element θ_{ij} signifies the link weight between agent i and agent j. These weights will be used for averaging the agents parameters. This matrix is computed offline based on the graph but the agents only need to know about (matrix elements of) their own neighborhoods. is a doubly stochastic matrix and $^{b}_{a}$ = 0 if (a, b) 2 E (Li et al., 2019). While the assumption of doubly stochastic communication matrix is very strict, these provide the powerful property of $\mathbf{x}$ = ... $\mathbf{x} = \frac{1}{N}\mathbf{I}$ $\mathbf{x}$, thus allowing for better convergence. Here, we also denote that $\mathbf{x}^\tau = [x_1^\tau, x_2^\tau, ..., x_N^\tau]^T$ is the vector of model parameters at iteration τ of the algorithm.

2.2 Minimum Connected Dominating Set

Connected Dominating Set (CDS) in a graph $G(V, E)$ is a connected set of nodes S V where each node has at least one connection to the nodes in this set. The CDS with minimum cardinality is the Minimum Connected Dominating Set (MCDS) ($\min_{S\ V} jSj$). Intuitively, if we consider the MCDS as one node in the graph, it connects and dominates all other nodes (yellow nodes in Figure 1).

Finding the MCDS is an NP-hard problem, and extensive studies have focused on proposing nearly exact approximation techniques to compute it. In this work, as a first step of our proposed method, we found the MCDS of the given graph using an approximation algorithm proposed by Butenko et al. (2004). This algorithm calculates the MCDS in $O(Nm)$ time complexity where N and m are the cardinalities of vertices and edges in the graph. This algorithm considers all vertices as nodes in the connected dominating set. Then, at each step, a node is selected using a greedy algorithm, and it is decided to either remove the node from the set or add it to the final solution S. Figure 2 shows the results of this greedy algorithm for four different graph topologies: Fully Connected (FC), Ring, Bipartite (Bipar), and a random graph.

3 Dominating Set Model Aggregation

3.1 The DSMA algorithm

We now describe our DSMA algorithm for DDL. The outline of our algorithm is shown in Figure 1 and the pseudocode is presented in Algorithm 1. First, given a graph topology $G(V, E)$, the MCDS (S) is

Algorithm 1 DSMA

Input: N, α , T, D_i,
Initialize x_i^0, v_i^0

Find MCDS S, and its corresponding and $\mathbf{W}$ matrices
for $\tau = 1 : T$ **do**
  {Perform following operations concurrently $\forall i \in V$}
  if $i \notin S$ **then**
    {Dominated update}
    $p_i^\tau = \mathbf{W}_i^i x_i^\tau + \sum_{l \in N_i^S} \mathbf{W}_i^l x_l^\tau$
  end if
  if $i \in S$ **then**
    {Dominating update}
    $p_i^\tau = \mathbf{W}_i^i x_i^\tau + \sum_{l \in N_i^O} \mathbf{W}_i^l x_l^\tau$
    $p_i^\tau = \sum_{l \in N_i^S} \quad {}_i^l p_l^\tau + \quad {}_i^i p_i^\tau$ {MCDS consensus}
  end if
  $x_i^{\tau+1} = p_i^\tau \quad \alpha g_i(x_i^\tau)$
end for

computed offline in a centralized manner, once in the beginning and further the resulting topology is used for training. The dominating nodes S constructs a smaller connected graph called $G(S, P)$, where $P \quad E$ is the set of edges from the original graph topology connecting the dominating nodes, S. Moreover, every agent $i \in V$ has a neighbor j in dominating set ($j \in S$) which we consider as *dominating neighbor* and the set of such neighbors denoted by N_i^S. Every agent $i \in V$ can also have a neighbor j which is not in the dominating set of whole graph ($j \in V \setminus S$), which we consider as *dominated neighbor*, and the set of such neighbors denoted by N_i^O. Formally, let $V = \{v_1, v_2, v_3, ..., v_n, r_1, r_2, ...r_n\}$ be the ordered set of all the *dominating agents* and all the *dominated agents* such that $S = \{v_1, v_2, ...v_n\}$ and $S = V \setminus S = \{r_1, r_2, ...r_n\}$, where $n = |S|$, $n = |S|$, and $N = n + n$.

After finding the MCDS and before explaining the our proposed algorithm, we also need to define a new communication matrix $\in \mathbb{R}^{n \quad n}$ where is also a doubly stochastic similar to . matrix represents the communication links among the dominating nodes; and, its elements are assigned based on the elements of the typical matrix in DDL literature (Li et al., 2019; Assran et al., 2019). This means if $\theta_{ij} \neq 0$ $(i, j \in S)$, then $\pi_{ij} \neq 0$, and otherwise if $\theta_{ij} = 0 (i, j \in S)$, then $\pi_{ij} = 0$. After considering non-zero elements as placeholders and based on Sinkhorn-Knopp algorithm (Sinkhorn, 1967), we can generate the doubly stochastic matrix. Formally, to generalize this matrix to the whole graph we can build up the $\mathbf{\Pi}$ matrix as

$$\mathbf{\Pi} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \end{bmatrix}, \tag{2}$$

where $\mathbf{I} \in \mathbb{R}^{n \quad n}$ and $\in \mathbb{R}^{n \quad n}$ as we defined. Basically, $\mathbf{\Pi}$ is a doubly stochastic matrix to ensure a consensus within the dominating agents. We also define another communication matrix $\mathbf{W} \in \mathbb{R}^{N \quad N}$ (in the form of block matrix) where

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{SS} & \mathbf{W}_{SS} \\ \mathbf{W}_{SS} & \mathbf{W}_{SS} \end{bmatrix}, \\ \mathbf{W}_{SS} = \mathrm{diag}(w_{r_1}^{r_1}, w_{r_2}^{r_2}, ..., w_{r_n}^{r_n}), \mathbf{W}_{SS} = \mathrm{diag}(w_{v_1}^{v_1}, w_{v_2}^{v_2}, ..., w_{v_n}^{v_n}), \mathbf{W}_{SS} = \mathbf{W}_{SS}^T, \tag{3}$$

$\mathbf{W}_{SS} \in \mathbb{R}^{n \quad n}$, $\mathbf{W}_{SS} \in \mathbb{R}^{n \quad n}$, $\mathbf{W}_{SS} \in \mathbb{R}^{n \quad n}$. Block $\mathbf{W}_{SS} = \left[w_i^j\right]_{n \quad n}$, $\forall i \in S, \forall j \in S$ is the matrix of weights corresponding to the information transmitted from the dominating agents S to the dominated agents S based on the specified graph topology $G(V, E)$ (the blue double sided arrows in Figure 1). Also, $\mathbf{W}_{SS}$ and $\mathbf{W}_{SS}$ matrices can be any diagonal matrix. Then, considering the non-zero elements of these matrices (blocks) as placeholders when constructing the $\mathbf{W}$ matrix, we can make $\mathbf{W}$ doubly stochastic based on the Sinkhorn-Knopp algorithm (Sinkhorn, 1967).

Intuitively, $\mathbf{\Pi}$ and $\mathbf{W}$ are two communication matrices which can give in effect similar behavior as . Now, based on our algorithm, the whole graph communication relation can be defined as $^0 = \mathbf{\Pi}\mathbf{W}$

(note that ^{0} also becomes doubly stochastic). Generally, $\mathbf{\Pi}$ and $\mathbf{W}$ of ^{0} in DSMA results in MCDS consensus and dominated nodes consensus, respectively. A unique feature of our algorithm is that the defined ^{0} matrix effectively prunes redundant connections of the graph $G(V, E)$ with communication matrix , leaving us with a new topology that efficiently shares information between different agents while reducing the communication overhead. Specifically, as a result of the product of ^{0} and the model parameters, the extra connections in the graph (connections between dominated nodes similar to the dotted lines in Figure1) will be eliminated for computing model parameters.

Once, $\mathbf{\Pi}$ and $\mathbf{W}$ are well defined for the MCDS, the DSMA training process begins (from step 2 of Algorithm 1). At each iteration τ, each dominated agent $i \in S$, collects the model parameters of its dominating neighbors (N_i^S) and aggregate them with its model parameter x_i^τ using weighted average. Concurrently, each dominating agent $i \in S$, weighted averages its dominated neighbors' (N_i^O) model parameters with its model parameter x_i^τ. These communication steps can be summarized using the matrix algebra as follows:

$$\mathbf{p}^\tau = \mathbf{W}\mathbf{x}^\tau, \tag{4}$$

Now, the dominating agents $i \in S$ have the whole graph model parameters, they can communicate with each other to find a consensus model parameters using the matrix ($\sum_{l \in N_i^S}$ $^{l}_{i}p_l^\tau +$ $^{i}_{i}p_i^\tau$). Here, p is the weighted average stored in the previous step. This consensus step can be formulated in a matrix form as follows:

$$\mathbf{p}^\tau = \mathbf{\Pi}\mathbf{p}^\tau, \tag{5}$$

Once a consensus is reached among the dominating agents in step 11 of the algorithm (equation 5), an SGD updating step is taken for all nodes $i \in V$ as follows:

$$\mathbf{x}^{\tau+1} = \mathbf{p}^\tau \quad \alpha\mathbf{g}(\mathbf{x}^\tau), \tag{6}$$

where $\mathbf{g}(\mathbf{x}^\tau) = [g_{v_1}(x^\tau), ..., g_{v_n}(x^\tau), g_{r_1}(x^\tau), ..., g_{r_n}(x^\tau)]^T$, and $g_i(x^\tau)$ is the computed gradient for agent i in iteration τ. We note that $\mathbf{p}^\tau = \mathbf{\Pi}\mathbf{W}\mathbf{x}^\tau$ where we denote $^{0} = \mathbf{\Pi}\mathbf{W}$ reflecting both the communication and the consensus steps.

3.2 Convergence Analysis

We now present our main convergence analysis considering the common assumptions in the DDL:

Assumption 1. *The following properties are assumed for matrix ^{0}:*
(1) $0 \quad \theta_{ij}^0 \quad 1$
(2) $\mathbf{I}^T \ ^{0} = \mathbf{I}^T$ *and* $^{0}\mathbf{1} = \mathbf{1}$ *(doubly stochastic)*
(3) If $i, j \in S = V \cap S$ *and* $i \neq j$, *then* $\theta_{ij}^0 = \theta_{ji}^0 = 0$
(4) $null\{\mathbf{I} \quad ^{0}\} = span\{1\}$.

Property 3 of the assumption 1 is a direct result of the proposed DSMA algorithm that removes the unnecessary communications between the dominated nodes and, hence, a reduction in the communication overhead. This property results from the structure of $\mathbf{W}$ and as discussed above. Also, since ^{0} is a doubly stochastic matrix, its eigenvalues is denoted as $1 = \lambda_1(\ ^{0}) > \lambda_2(\ ^{0})$ ... $\lambda_N(\ ^{0})$ 0.

Assumption 2. *(β-smoothness) For each agent i, function $f_i : \mathbb{R}^d \to \mathbb{R}$ should be β_i-smooth which means it is continuously differentiable and for all $x, y \in \mathbb{R}^d$ we have:*

$$f_i(y) \quad f_i(x) \quad \nabla f_i(x)^T(y \quad x) \quad \frac{\beta_i}{2}\|y \quad x\|^2 \tag{7}$$

Assumption 3. *(κ-strongly convex) The objective functions $f_i : \mathbb{R}^d \to \mathbb{R}$ are assumed to be κ_i strongly convex, i.e. for all $x, y \in \mathbb{R}^d$, we have:*

$$f_i(y) \quad f_i(x) \quad \nabla f_i(x)^T(y \quad x) \quad \frac{\kappa_i}{2}\|y \quad x\|^2 \tag{8}$$

Assumption 4. *The objective functions $f_i : \mathbb{R}^d \to \mathbb{R}$ are considered proper (not everywhere infinite), and coercive which means; if $\|x\| \to 1$, then $f(x) \to 1$.*

Based on assumptions 2, 3, and 4, we can conclude that the gradient of function f_i is Lipschitz continuous which means $\|\nabla f_i(y) \quad \nabla f_i(x)\| \quad \beta_i\|y \quad x\|, \forall x, y \in \mathbb{R}^d$.Therefore, considering the whole agents objective functions, $\sum_{i=1}^N f_i(x_i)$ is strongly convex with $\kappa = \min \kappa_i$ and its gradient has Lipschitz parameter $\beta = \max \beta_i$. Also, $\kappa \quad \beta$ based on the strong convexity and Lipschitz properties of ∇f_i.

3.2.1 Intermediate Concepts

Before providing the theoretical results, here we explain some concepts which will help in proof of the algorithm convergence. More details about the proof are provided in the Appendix section. In order to prove the convergence of this algorithm we use the Lyapanov Stability method and define a Lyapanov function. Let us consider equation 6, and re-write this equation by adding and subtracting $\mathbf{x}^\tau$ and considering $= \mathbf{I}$ ^{0}, we will have:

$$\mathbf{x}^{\tau+1} = \mathbf{x}^\tau \quad \mathbf{x}^\tau + {}^0\mathbf{x}^\tau \quad \alpha \mathbf{g}(\mathbf{x}^\tau) = \mathbf{x}^\tau \quad \alpha(\mathbf{g}(\mathbf{x}^\tau) + \alpha^{\ 1} \mathbf{x}^\tau) \tag{9}$$

Considering 9, updating of the model parameters happens through $r F = \mathbf{g}(\mathbf{x}^\tau) + \alpha^{\ 1} \mathbf{x}^\tau$. Therefore, equation 9 can be rewritten as $\mathbf{x}^{\tau+1} = \mathbf{x}^\tau \quad \alpha r F$ where $r F$ is, in fact, the effective gradient after applying the algorithm steps.

We note that $r F$ is the Stochastic Lyapanov Gradient (Zeng and Yin, 2018). Assuming $\mathbf{F}(\mathbf{x}) = [f_{v_1}(x^\tau), ..., f_{v_n}(x^\tau), f_{r_1}(x^\tau), ..., f_{r_n}(x^\tau)]^T$ and $\mathbf{1}$ 2 R^N, the appropriate Lyapanov function can be proposed as the following:

$$V(\mathbf{x}, \alpha) := \frac{N}{n}\mathbf{1}^T\mathbf{F}(\mathbf{x}) + \frac{1}{2\alpha}k\mathbf{x}k^2 , \tag{10}$$

where $k\mathbf{x}k^2$ is the norm of $\mathbf{x}$ with respect to . (The $\frac{1}{2\alpha}k\mathbf{x}k^2$ is similar to adding penalization (Frobenius norm) to our algorithm).

As we mentioned earlier, $\sum_{i=1}^N f_i(x_i)$ is strongly convex with $\kappa = \min \kappa_i$ and its gradient has Lipschitz parameter $\beta = \max \beta_i$, the gradient of $r V(\mathbf{x})$ is Lipschitz continuous (it is continuously differentiable). The Lipschitz constant for $r V(\mathbf{x})$ is $\beta = \beta + \alpha^{\ 1}\lambda_{max}(\ ) = \beta + \alpha^{\ 1}(1 \ \lambda_N(\ )$. Similarly, it can be concluded that $V(\mathbf{x})$ is also strongly convex with parameter $\varkappa = \kappa + (2\alpha)^{\ 1}\lambda_{min}(\ ) = \kappa + (2\alpha)^{\ 1}((1 \quad \lambda_2(\ ))$. It is worth mentioning that, following from $\kappa \quad \beta$, we can conclude that $\varkappa \quad \beta$. Based on the strong convexity property of Lyapanov function V, its difference with the optimum value is bounded for $8\mathbf{x}$ 2 R^N as follows:

$$2\varkappa(V(\mathbf{x}) \quad V^*) \quad k r V(\mathbf{x})k^2. \tag{11}$$

Considering this property of $V(\mathbf{x})$, the main theoretical convergence result can be determined. First we introduce a lemma that shows implementing the proposed DSMA algorithm leads to sufficient decrease in the Lyapunov function.

Lemma 1. *Under Assumptions 1 - 4, the following holds true for each iteration τ 2 N of DSMA:*

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad V(\mathbf{x}^\tau) \qquad \alpha r V(\mathbf{x}^\tau)^T\mathsf{E}[r F_i(\mathbf{x}^\tau)] + \frac{\beta}{2}\alpha^2\mathsf{E}[k r F_i(\mathbf{x}^\tau)k^2]. \tag{12}$$

Moreover to ensure convergence of the proposed algorithm, we utilize a standard assumption in the context of (centralized) deep learning on the characteristics of $F_i(\mathbf{x}^\tau)$.

Assumption 5. *For all τ 2 N, scalars $c_2 \quad c_1 > 0$, and $\delta_1, \delta_2 \quad 0$ can be set such that:*
(a) $r V(\mathbf{x}^\tau)^T\mathsf{E}[r F_i(\mathbf{x}^\tau)] \quad c_1 k r V(\mathbf{x}^\tau)k^2$,
and $k\mathsf{E}[r F_i(\mathbf{x}^\tau)]k \quad c_2 k r V(\mathbf{x}^\tau)k$,
(b)$Var[r F_i(\mathbf{x}^\tau)] \quad \delta_1 + \delta_2 k r V(\mathbf{x}^\tau)k^2$.

We note that Assumption 5(a) guarantees that sufficient descent of V happens in the direction of $F_i(\mathbf{x}^\tau)$. Based on Assumption 5(b), the variance of $r F_i(\mathbf{x}^\tau)$ is bounded above by the second moment of $r V(\mathbf{x}^\tau)$. Expanding the variance of $r F_i(\mathbf{x}^\tau)$, the second moment of $r F_i(\mathbf{x}^\tau)$ can be bounded above as

$$\mathsf{E}[k r F_i(\mathbf{x}^\tau)k^2] \quad \delta_1 + \delta k r V(\mathbf{x}^\tau)k^2, \tag{13}$$

Where $\delta := \delta_2 + c_2^2 \quad c_1^2 > 0$. Considering Assumption 5, Lemma 1 can be re-organized as follows:

Lemma 2. *Under Assumptions 1- 5, each iteration τ 2 N of the proposed DSMA algorithm satisfies:*

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad V(\mathbf{x}^\tau) \qquad (c_1 \quad \frac{\beta}{2}\alpha\delta)\alpha k r V(\mathbf{x}^\tau)k^2 + \frac{\beta}{2}\alpha^2\delta_1, \tag{14}$$

where the step size (α) is $0 < \alpha \quad \frac{c_1}{\beta\delta}$ to impose sufficient condition for the rest of analysis.

Table 2: Test accuracy and communication time (per epoch) for CNN model training on CIFAR10.

			Accuracy (%)					Communication time (s)				
	Number of agents		5	10	30	60	100	5	10	30	60	100
Communication-efficient methods	DPMSGD (Lian et al., 2017)	FC	76.3	74.6	70.5	68.8	59.0	3.1	5.7	13.5	32.5	45.6
		Ring	75.7	74.0	66.7	59.8	56.3	3.1	5.5	14.4	31.4	44.6
		Bipar	75.3	74.5	64.9	52.5	44.1	3.2	5.3	14.1	31.7	45.5
	SGP (Assran et al., 2019)	FC	71.7	71.04	59.2	46.6	29.3	3.6	5.4	13.9	32.3	46.2
		Ring	70.9	71.2	59.1	46.9	29.3	3.1	5.3	13.8	31.5	45.0
		Bipar	70.9	71.0	59.8	47.5	29.4	4.2	5.4	13.8	31.5	44.4
	Matcha (Wang et al., 2019)	FC	60.9	55.3	-	-	-	1.5	2.4	-	-	-
		Ring	60.7	55.7	-	-	-	1.8	2.8	-	-	-
		Bipar	54.1	54.1	-	-	-	1.7	2.7	-	-	-
	LDSGD (Li et al., 2019)	FC	**76.3**	**74.4**	**68.7**	**66.5**	**58.6**	2.1	3.8	8.6	20.6	30.4
		Ring	75.5	73.0	65.2	58.7	54.4	2.1	3.6	9.6	20.9	29.7
		Bipar	75.6	74.5	63.5	50.6	43.9	2.2	3.5	9.4	21.2	30.3
	SwarmSGD (Nadiradze et al., 2021)	FC	75.3	72.3	50.6	27.8	12.3	1.7	2.8	6.5	16.7	22.2
		Ring	75.1	72.7	49.9	25.7	10.9	**1.6**	**2.6**	**7.2**	**16.5**	**23.3**
		Bipar	**75.9**	71.4	51.0	28.9	12.9	1.5	2.7	7.1	16.7	22.4
	DSMA	FC	74.7	72.5	67.5	64.5	55.8	**0.7**	**0.7**	**0.5**	**0.5**	**0.4**
		Ring	**75.2**	**73.1**	**65.4**	**60.1**	**56.6**	1.7	3.9	13.5	30.6	43.7
		Bipar	74.6	**71.8**	**64.9**	**60.2**	**52.9**	**1.1**	**0.9**	**1.0**	**1.0**	**0.9**

Next, in Appendix we show that for a general loss function, the proposed DSMA algorithm will lead to consensus among the agents if the step size does not exceed the specified upper limit in Lemma 2 . All the proofs are deferred to Appendix. Our main theorem shows that iterates of the DSMA algorithm converge to a neighborhood of the optimal value.

Theorem 1. *(Convergence of DSMA) Under assumptions 1-5, the following inequality holds for each iterate τ 2 N of DSMA algorithm for the strongly convex case:*

$$\lim_{\tau \; ! \; 1} \mathsf{E}[V(\mathbf{x}^{\tau}) \quad V \;] \quad \frac{\alpha\beta\delta_1}{2\kappa c_1}$$

$$\mathsf{E}[V(\mathbf{x}^{\tau}) \quad V \;] \quad (1 \quad \alpha\kappa c_1)^{\tau \;\; 1}(V(\mathbf{x}^1) \quad V \;) + \frac{\alpha^2\beta\delta_1}{2}\sum_{l=0}^{\tau \;\; 1}(1 \quad \alpha\kappa c_1)^l, \tag{15}$$

where V is the optimal value, and α satisfies $0 < \alpha \quad \frac{c_1 \;\; (1 \;\; \lambda_N(\;\; ^{\theta}))\delta}{\beta\delta}$.

Based on this theorem we conclude that $V(x)$ is linearly converging to a neighborhood of the optimal value ($\lim_{\tau \; ! \; 1} \mathsf{E}[V(\mathbf{x}^{\tau}) \quad V \;] \quad \frac{\alpha\beta\delta_1}{2\kappa c_1}$). Proofs and DSMA convergence result for the non-convex case are presented in the Appendix.

4 Experimental Results

In this section, the experimental results for the DSMA in terms of convergence, accuracy, scalability, and communication time are presented. Moreover, the performance of our algorithm is compared with benchmark algorithms in decentralized deep learning such as: SGP (Assran et al., 2019), momentum variant of DPSGD or DPMSGD (Lian et al., 2017), LDSGD (Li et al., 2019), Matcha (Wang et al., 2019), and SwarmSGD (Nadiradze et al., 2021).

Empirical studies are conducted for 5, 10, 30, 60, and 100 agents to evaluate the scalability of our algorithm. The training is applied to two benchmark datasets; MNIST and CIFAR10. Our method is examined for two different model architectures: a deep convolutional neural network (CNN) (LeCun et al., 1998) and ResNet20 (He et al., 2016). For training, the mini-batch is selected with the batch size set as 128. The 0.01 step size is initially used and is decayed with the constant 0.98. The momentum constant in all of the experiments is considered 0.9. After training, the average accuracy of the models for training and testing data is reported in plots and tables (the results for the MNIST dataset with the CNN model architecture and CIFAR10 with the ResNet20 model architecture are reported in the Appendix). Each experiment was run three times, and its average value is reported. The code is written with the PyTorch distributed package. Our presented results are executed using a high-performance cluster with 15 nodes and a total of 60 Nvidia A100 80GB GPUs.

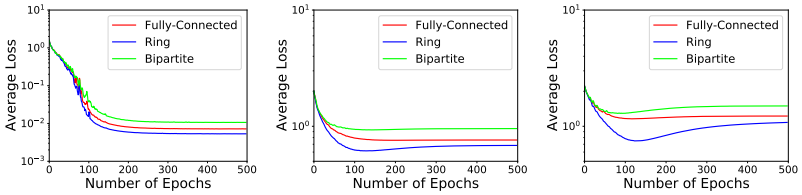


Figure 3: Convergence comparison (average training loss) for different graph topologies for training (a) 5, (b) 30, and (c) 100 agents.

4.1 Model Convergence

To verify the correctness of the convergence analysis, we evaluate the convergence characteristic of the loss function for a different number of agents in Figure 3 and for different dense levels for graph topologies (similar to previous studies (Tang et al., 2018)), where Fully Connected represents a dense graph, and Ring and Bipartite are sparser graphs but with different connectivity protocols (as shown in Figure 2). Among these graphs, Fully connected and Ring are r-regular, and Bipartite graph is not r-regular. Also, these graph topologies can represent different scenarios in evaluating our method by containing 1, 2 and V 2 nodes in their MCDS, respectively.

Although by increasing the number of agents the convergence rate of the proposed method is hurt similar to other DDL methods for all graph topologies, DSMA finally converges to a specific loss value. Moreover, it can be observed that the ring topology has better convergence. This phenomenon may arise because the MCDS consensus step in our algorithm has more connected elements for ring topology than Bipartite and Fully Connected graphs (The number of agents in MCDS for Ring topology is always V 2). On the other hand, in the comparison section, we will show that more connectivity results in more communication (Table 2).

4.2 Comparative Analysis

In this section, we compare our DSMA algorithm with other benchmark DDL methods. Some considerations for comparison are: (i) As the best communication is reported in the LDSGD paper for ten local SGD steps versus 20 collaborative steps, the same assumption is considered for producing their results. (ii) The graphs in SGP optimizer are considered undirected. (iii) In the Matcha method, the communication budget is considered as 0.5. Also, since the graph matching sets were not implemented for large-scale networks, the results for Matcha are only shown for 5 and 10 agents.

Table 2 shows the test accuracy and agents total communication time of each method for different graph topologies. Bold values represent the best accuracy and communication time among the communication-efficient methods. Moreover, results for different number of agents are reported to indicate the scalability of algorithms. The test accuracy for SwarmSGD and SGP methods drop as the number of agents increases. This trend is also more observable through Figure 4 (a)-(c) and (e)-(g) which plotted test accuracy results of different methods for large-scale networks (60 and 100 agents). These results are expected and discussed previously in (Assran et al., 2019; Sattler et al., 2019). Results in Figure 4 are presented for the extremely dense graph (Fully Connected) with only one node in MCDS as well as two sparse graphs to compare the two different cases of having a high (V 2 for Ring) and a low number of nodes (2 for Bipartite) in MCDS.

While, the DSMA algorithm shows an acceptable performance in terms of scalibility compared to previous studies in different graph topologies, it significantly outperforms others in the case of a Bipartite graph (Figure 4). This behaviour shows two significant characteristics of DSMA algorithm: (i) the MCDS consensus step is critical in efficiently propagating information in the graph and since the Fully Connected graph always has one node in its MCDS, this propagation of information is indeed hurt. (ii) Apart from the case of Fully Connected graphs without any nodes in its MCDS, if fewer nodes participate in MCDS consensus step, more redundant connections will be removed which results in higher accuracies and lower communication rates (Table 2). This behaviour can be clearly observed in the case of a Bipartite graph with two nodes in its MCDS. The DSMA algorithm also converges smoother than communication-efficient methods with local updates (e.g., LDSGD).

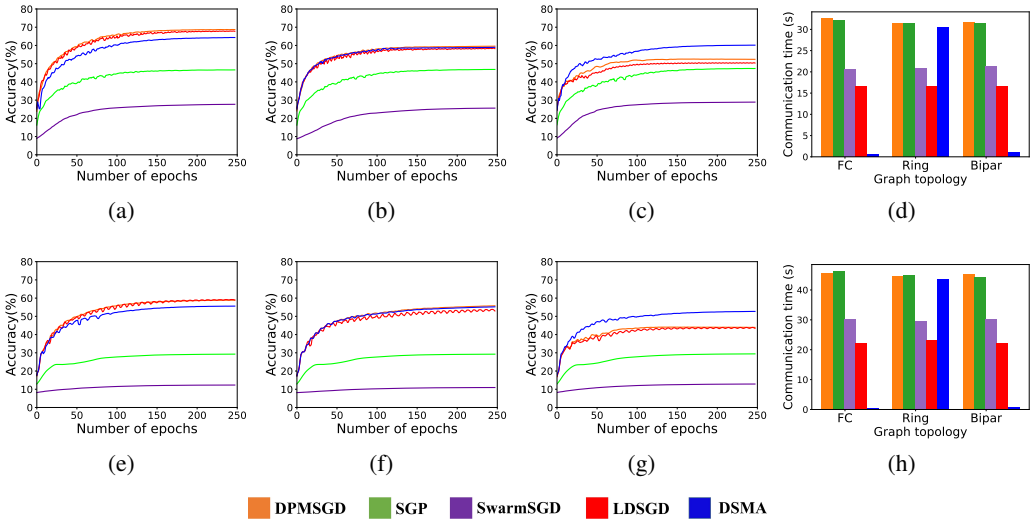


Figure 4: Average test accuracy for 60 agents in (a) FC, (b) Ring, and (c) Bipartite graphs, and (d) communication time per epoch for 60 agents; average test accuracy for 100 agents in (e) FC, (f) Ring, (g) Bipartite graphs, and (h) communication time per epoch for 100 agents.

To visually illustrate communication reduction and methods scalability, Figure 4 (d) and Figure 4 (h) shows communication time comparison for different graph topologies in large-scale networks (60, and 100 agents). Communication time is considered as the time it takes from agents to send their model parameters to their correspondence neighbors per epoch (averaged over 30 epochs). Primary observation from Figure 4 (d) and Figure 4 (h) is that in the Fully Connected and Bipartite cases, the communication time for DSMA drops dramatically compared to other methods. This happens because fewer nodes (nodes in MCDS) need to gather model parameters from their neighbors and then communicate with each other to achieve consensus. Additionally, as we explained, the model performance also exceeds other methods for large-scale networks in the Bipartite case (Figure 4). In Ring graph topology, since the number of agents in MCDS is V 2, the reduction in communication time is negligible. Nevertheless, for the same reason, its accuracy is very close to other methods (e.g., DPMSGD). In general, as it is explained above, our algorithm considers the edges between the dominated nodes and the dominating nodes as communication bridges for aggregating the model parameters, and the edges between the dominated nodes are considered negligible. This in turn results in a high reduction of communication overhead in Fully Connected and Bipartite cases where the number of edges between the dominated nodes are high.

5 Conclusion

In this paper, we propose a new algorithm called Minimum Connected Dominating Set Model Aggregation (DSMA) to reduce the communication overhead in decentralized deep learning. The method is empirically examined over different graph topology networks (dense and sparse graphs) on two benchmark datasets and two network architectures (three layers CNN and Resnet20). Also, the scalability of DSMA is examined through experimental analysis. Results show that even in large-scale networks, the communication rate reduces dramatically while preserving comparable test accuracy. We believe DSMA can be implemented on top of other decentralized algorithms to reduce their communication time which we consider this path as future work.

Acknowledgments and Disclosure of Funding

This work was partly supported by the National Science Foundation under grants CAREER-1845969 and COALESCE-1954556.

References

Assran, M., N. Loizou, N. Ballas, and M. Rabbat (2019). Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pp. 344–353. PMLR.

Butenko, S., X. Cheng, C. A. Oliveira, and P. M. Pardalos (2004). A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks. In *Recent developments in cooperative control and optimization*, pp. 61–73. Springer.

Esfandiari, Y., K. Nagasubramanian, F. Fotouhi, P. S. Schnable, B. Ganapathysubramanian, and S. Sarkar (2021). Distributed deep learning for persistent monitoring of agricultural fields. In *NeurIPS 2021 AI for Science Workshop*.

Esfandiari, Y., S. Y. Tan, Z. Jiang, A. Balu, E. Herron, C. Hegde, and S. Sarkar (2021). Cross-gradient aggregation for decentralized learning from non-iid data. In *International Conference on Machine Learning*, pp. 3036–3046. PMLR.

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Kairouz, P., H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. (2019). Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Kempe, D., A. Dobra, and J. Gehrke (2003). Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pp. 482–491. IEEE.

Koloskova, A., N. Loizou, S. Boreiri, M. Jaggi, and S. Stich (2020). A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pp. 5381–5393. PMLR.

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

Li, M., D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su (2014). Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 583–598.

Li, X., W. Yang, S. Wang, and Z. Zhang (2019). Communication-efficient local decentralized sgd methods. *arXiv preprint arXiv:1910.09126*.

Lian, X., C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu (2017). Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems 30*.

Lu, Y. and C. De Sa (2020). Moniqua: Modulo quantized communication in decentralized sgd. *arXiv preprint arXiv:2002.11787*.

McMahan, B., E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR.

Nadiradze, G., A. Sabour, P. Davies, S. Li, and D. Alistarh (2021). Asynchronous decentralized sgd with quantized and local updates. *Advances in Neural Information Processing Systems 34*, 6829–6842.

Sattler, F., S. Wiedemann, K.-R. Müller, and W. Samek (2019). Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems 31*(9), 3400–3413.

Scaman, K., F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee (2018). Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pp. 2740–2749.

Sinkhorn, R. (1967). Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly 74*(4), 402–405.

Sun, Y., H. Ochiai, and H. Esaki (2021). Decentralized deep learning for mobile edge computing: A survey on communication efficiency and trustworthiness. *arXiv preprint arXiv:2108.03980*.

Tang, H., X. Lian, M. Yan, C. Zhang, and J. Liu (2018). $d^2$2: Decentralized training over decentralized data. In *International Conference on Machine Learning*, pp. 4848–4856. PMLR.

Tang, Z., S. Shi, and X. Chu (2020). Communication-efficient decentralized learning with sparsification and adaptive peer selection. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1207–1208. IEEE.

Vogels, T., S. P. Karimireddy, and M. Jaggi (2020). Practical low-rank communication compression in decentralized deep learning. *Advances in Neural Information Processing Systems 33*.

Wang, B., Y. Sun, T. Do-Duy, E. Garcia-Palacios, and T. Q. Duong (2021). Adaptive d-hop connected dominating set in highly dynamic flying ad-hoc networks. *IEEE Transactions on Network Science and Engineering 8*(3), 2651–2664.

Wang, J., A. K. Sahu, Z. Yang, G. Joshi, and S. Kar (2019). Matcha: Speeding up decentralized sgd via matching decomposition sampling. In *2019 Sixth Indian Control Conference (ICC)*, pp. 299–300. IEEE.

Wu, J. and H. Li (1999). On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pp. 7–14.

Ying, B., K. Yuan, Y. Chen, H. Hu, P. Pan, and W. Yin (2021). Exponential graph is provably efficient for decentralized deep training. *Advances in Neural Information Processing Systems 34*.

Yu, H., R. Jin, and S. Yang (2019). On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, pp. 7184–7193. PMLR.

Yu, J., N. Wang, G. Wang, and D. Yu (2013). Connected dominating sets in wireless ad hoc and sensor networks–a comprehensive survey. *Computer Communications 36*(2), 121–134.

Zeng, J. and W. Yin (2018). On nonconvex decentralized gradient descent. *IEEE Transactions on signal processing 66*(11), 2834–2848.

A Appendix

A.1 Additional Results and Proofs

Following from the lemmas and theorems in the manuscript, this Appendix section focuses on proving them. We first prove the Lemmas and then using the proved lemmas we prove two Theorems for convergence of DSMA in convex and non-convex cases.

Lemma 1. *Under Assumptions 1 - 4, the following holds true for each iteration* $\tau \in \mathbb{N}$ *of the proposed DSMA algorithm:*

$$\mathbb{E}[V(\mathbf{x}^{\tau+1})] - V(\mathbf{x}^{\tau}) \leq -\gamma \nabla V(\mathbf{x}^{\tau})^T \mathbb{E}[\nabla F_i(\mathbf{x}^{\tau})] + \frac{\tilde{\gamma}^2 \beta}{2} \mathbb{E}[\|\nabla F_i(\mathbf{x}^{\tau})\|^2]; \tag{16}$$

Proof. Based on the β-smoothness assumption (Assumption 2), the iterates generated by DSMA satisfy:

$$V(\mathbf{x}^{\tau+1}) - V(\mathbf{x}^{\tau}) \leq \nabla V(\mathbf{x}^{\tau})^T(\mathbf{x}^{\tau+1} - \mathbf{x}^{\tau}) + \frac{\beta}{2}\|\mathbf{x}^{\tau+1} - \mathbf{x}^{\tau}\|^2 \tag{17}$$

Then, based on the definition of $\nabla F_i(\mathbf{x}^{\tau})$ and equation 10, and by taking expectations from the inequality, we can obtain:

$$\mathbb{E}[V(\mathbf{x}^{\tau+1})] - V(\mathbf{x}^{\tau})] \leq \mathbb{E}[-\gamma \nabla V(\mathbf{x}^{\tau})^T \nabla F_i(\mathbf{x}^{\tau}) + \frac{\tilde{\gamma}^2 \beta}{2}\|\nabla F_i(\mathbf{x}^{\tau})\|^2] \tag{18}$$

Due to the random sampling aspect, $V(\mathbf{x}^{\tau})$ is deterministic, $V(\mathbf{x}^{\tau+1})$ is stochastic. Therefore, $\mathbb{E}[V(\mathbf{x}^{\tau})] = V(\mathbf{x}^{\tau})$ and we can complete the proof as follows:

$$\mathbb{E}[V(\mathbf{x}^{\tau+1})] - V(\mathbf{x}^{\tau}) \leq -\gamma \nabla V(\mathbf{x}^{\tau})^T \mathbb{E}[\nabla F_i(\mathbf{x}^{\tau})] + \frac{\tilde{\gamma}^2 \beta}{2} \mathbb{E}[\|\nabla F_i(\mathbf{x}^{\tau})\|^2] \tag{19}$$

□

Lemma 2. *Under Assumptions 1-5, each iteration* $\tau \in \mathbb{N}$ *of the proposed DSMA algorithm satisfies the following:*

$$\mathbb{E}[V(\mathbf{x}^{\tau+1})] - V(\mathbf{x}^{\tau}) \leq (c_1 - \frac{\tilde{\gamma}}{2}) \|\nabla V(\mathbf{x}^{\tau})\|^2 + \frac{\tilde{\gamma}}{2}\zeta_1; \tag{20}$$

where the step size (γ) *is* $0 < \gamma \leq \frac{c_1}{\beta\delta}$ *to impose sufficient condition for the rest of analysis.*

Proof. Considering Lemma 1 and the first part of Assumption 5, we have

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad V(\mathbf{x}^{\tau}) \qquad c_1 \quad k\, r\, V(\mathbf{x}^{\tau})k^2 + \frac{\tilde{}}{2} \quad {}^2\mathsf{E}[k\, r\, F_i(\mathbf{x}^{\tau})k^2] \tag{21}$$

Then, based on the second part of assumption 5, we can obtain:

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad V(\mathbf{x}^{\tau}) \qquad c_1 \quad k\, r\, V(\mathbf{x}^{\tau})k^2 + \frac{\tilde{}}{2} \quad {}^2(\ {}_1 + \quad k\, r\, V(\mathbf{x}^{\tau})k^2) \tag{22}$$

By considering Equation 14, we can complete the proof as follows:

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad V(\mathbf{x}^{\tau}) \qquad (c_1 \quad \frac{\tilde{}}{2} \quad) \quad k\, r\, V(\mathbf{x}^{\tau})k^2 + \frac{\tilde{}}{2} \quad {}^2 \ {}_1 \tag{23}$$

□

Before proving Proposition 1, first we need to state some technical lemmas.

Lemma 3. *V has a lower bound denoted by V_{inf} over an open set which contains the iterates $\{\mathbf{x}^{\tau}\}$ generated by DSMA (Algorithm 1).*

Lemma 3 can be concluded since f_i is proper and coercive based on assumption 4.

Lemma 4. *Let Assumption 1-4 holds. There exists an upper bound for the gradient expected value with a constant $0 < U < 1$ as $\mathsf{E}[k\mathbf{g}(\mathbf{x}^{\tau})k] \quad U$.*

The proof of this lemma directly follows from the Assumption 2 and 3 (Lipschitz continuity) and $U = \max_i U_i$.

Proposition 1. *Let assumptions 1-4 hold. Then the following is stated for each iteration of DSMA algorithm:*

$$\mathsf{E}[kx_i^{\tau} \quad a^{\tau}k] \quad \frac{U}{1 \quad {}_2(\ {}^0)} \quad 8 \quad 2\ \mathsf{N}; \tag{24}$$

where a^{τ} is the average of model parameters for different agents ($a^{\tau} = \frac{1}{N}\sum_{i=1}^{N} x_i^{\tau}$) and U is an upper bound of $\mathsf{E}[k\mathbf{g}(\mathbf{x}_i)k]$; $8 \quad 2\ \mathsf{N}$ as discussed in Lemma 4 and satisfies $0 < \frac{c_1 \quad (1 \quad \lambda_N(\ ))\delta}{\beta\delta}$.

Proof. From the DSMA algorithm we have:

$$\mathbf{x}^{\tau+1} = \quad {}^0\mathbf{x}^{\tau} \qquad \mathbf{g}(\mathbf{x}^{\tau}); \tag{25}$$

We then can get the following:

$$\mathbf{x}^{\tau} = \quad \sum_{t=0}^{\tau \quad 1} \quad {}^{0\tau \quad 1 \quad t}\mathbf{g}(\mathbf{x}^t); \tag{26}$$

Now letting $\mathbf{a}^{\tau} = [a^{\tau}; a^{\tau}; :::; a^{\tau}]^T \ 2\ \mathsf{R}^N$, we have:

$$\mathbf{a}^{\tau} = \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T\mathbf{x}^{\tau}; \tag{27}$$

The following derivations can be obtained:

$$\begin{aligned}
\mathsf{E}[kx_i^{\tau} \quad a^{\tau}k] \quad & \mathsf{E}[k\mathbf{x}^{\tau} \quad \mathbf{a}^{\tau}k] \\
&= \mathsf{E}[k \quad \sum_{t=0}^{\tau \quad 1} \quad {}^{0\tau \quad 1 \quad t}\mathbf{g}(\mathbf{x}^t) + \quad \sum_{t=0}^{\tau \quad 1} \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T \quad {}^{0\tau \quad 1 \quad t}\mathbf{g}(\mathbf{x}^t)k] \\
&= \mathsf{E}[k \quad \sum_{t=0}^{\tau \quad 1} \quad {}^{0\tau \quad 1 \quad t}\mathbf{g}(\mathbf{x}^t) + \quad \sum_{t=0}^{\tau \quad 1} \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T\mathbf{g}(\mathbf{x}^t)k] \\
&= \quad \mathsf{E}[k\sum_{t=0}^{\tau \quad 1}(\ {}^{0\tau \quad 1 \quad t} \quad \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T)\mathbf{g}(\mathbf{x}^t)k] \\
&\quad \sum_{t=0}^{\tau \quad 1}\mathsf{E}[k \quad {}^{0\tau \quad 1 \quad t} \quad \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^Tk] \quad \mathsf{E}[k\mathbf{g}(\mathbf{x}^t)k] \\
&= \quad \sum_{t=0}^{\tau \quad 1} \quad {}_2(\ {}^0)^{\tau \quad 1 \quad t}\mathsf{E}[k\mathbf{g}(\mathbf{x}^t)k];
\end{aligned} \tag{28}$$

where we used the triangle inequality, and properties of norm operation and double stochastic matrix ^{0}. From Lemma 4 $\mathsf{E}[k\mathbf{g}(\mathbf{x}^t)k]$ U and $_2(\ ^0) < 1$, thus we have:

$$\begin{aligned}\mathsf{E}[kx_i^\tau \quad a^\tau k] \quad & \sum_{t=0}^{\tau\ 1} {}_2(\ ^0)^{\tau\ 1\ t}\mathsf{E}[k\mathbf{g}(\mathbf{x}^t)k] \\ & \sum_{t=0}^{\tau\ 1} {}_2(\ ^0)^{\tau\ 1\ t}U \\ & \frac{U}{1\quad {}_2(\ ^0)}\end{aligned} \tag{29}$$

which completes the proof. □

Now, let us recall the main theroem of this paper for strongly convex case and prove it.

Theorem 1. *(Convergence of DSMA)* Under assumptions 1-5, the following inequality holds for each iterate 2 N of DSMA algorithm for the strongly convex case:

$$\begin{aligned}\mathsf{E}[V(\mathbf{x}^\tau) \quad V\ ] \quad & (1 \quad \sim c_1)^{\tau\ 1}(V(\mathbf{x}^1) \quad V\ ) \\ & + \frac{^{2\sim}{}_1}{2}\sum_{l=0}^{\tau\ 1}(1 \quad \sim c_1)^l;\end{aligned} \tag{30}$$

where V *is the optimal value, and* *satisfies* $0 <$ $\frac{c_1\ \ (1\ \ \lambda_N(\ ^\theta))\delta}{\beta\delta}$.

Proof. Let us consider Lemma 2 and Equation 11 from the main manuscript. Then, we obtain

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad V(\mathbf{x}^\tau) \qquad (c_1 \quad \frac{\sim}{2}\ ) \ \ k\,r\,V(\mathbf{x}^\tau)k^2 + \frac{\sim}{2}{}^{2}{}_1 \qquad \frac{1}{2}\ c_1 k\,r\,(\mathbf{x}^\tau)k^2 + \frac{^{2\sim}{}_1}{2} \tag{31}$$

Then, follows from $\frac{c_1}{\beta\delta}$ (extracted from $\frac{c_1\ \ (1\ \ \lambda_N(\ ^\theta))\delta}{\beta\delta}$), we can obtain:

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad V(\mathbf{x}^\tau) \qquad \frac{1}{2}\ c_1 k\,r\,(\mathbf{x}^\tau)k^2 + \frac{^{2\sim}{}_1}{2} \tag{32}$$

Also, based on assumption 3, we can rewrite the inequality as follows:

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad V(\mathbf{x}^\tau) \qquad c_1{\sim}(V(\mathbf{x}^\tau) \quad V\ ) + \frac{^{2\sim}{}_1}{2} \tag{33}$$

After recursively taking expectation and subtracting V from both sides of the inequality, the following inequality can obtain:

$$\mathsf{E}[V(\mathbf{x}^{\tau+1}) \quad V\ ] \quad (1 \quad \sim c_1)\mathsf{E}[V(\mathbf{x}^\tau) \quad V\ ] + \frac{^{2\sim}{}_1}{2} \tag{34}$$

As $0 <$ $\sim c_1$ $\frac{\kappa c_1^2}{\beta\delta}$ $\frac{\kappa c_1^2}{\beta c_1^2} = \frac{\kappa}{\beta}$ 1, the conclusion follows by applying Equation 34 recursively through iterations 2 N. □

Theorem 2: Let Assumptions 1-5 hold and the learning rate satisfies the following inequality for 8 2 N:

$$0 < \qquad \frac{c_1 \quad (1 \quad {}_N(\ ^0))}{\sim}$$

The DSMA iterates for non-convex case can satisfy the following inequality:

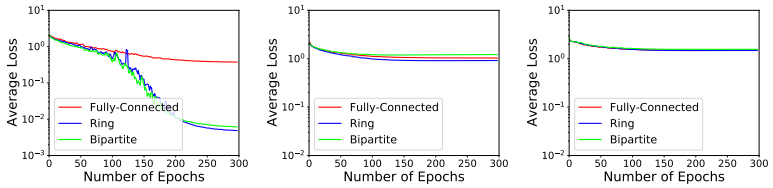
$$\mathsf{E}[\sum_{\tau=1}^{m} k\,r\,V(\mathbf{x}^\tau)k^2] \quad \frac{\sim m\quad 1}{c_1} + \frac{2(V(\mathbf{x}_1) \quad V_{\inf})}{c_1} = \frac{(\sim\ + 1 \quad {}_N(\ ^0))m\ {}_1}{c_1} + \frac{2(V(\mathbf{x}_1) \quad V_{\inf})}{c_1} \tag{35}$$

Proof. Let again consider Lemma 2, and take the expectation from both sides of the inequality,

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad \mathsf{E}[V(\mathbf{x}^\tau)] \qquad (c_1 \quad \frac{\sim}{2}) \ \ \mathsf{E}[k\,r\,V(\mathbf{x}^\tau)k^2] + \frac{\sim\ ^2\ {}_1}{2} \tag{36}$$

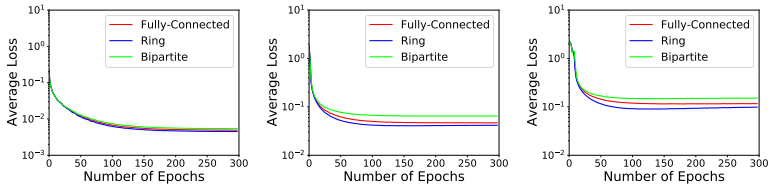
Table 3: Time (ms) needed for computing Minimum Connected Dominating Set

Number of agents	5	10	30	60	100
Fully Connected	5.3	6.5	19.9	77.4	112.7
Ring	5.0	6.1	12.1	56.8	77.5
Bipartite	5.2	6.7	15.1	57.0	80.5



(a) (b) (c)

Figure 5: Convergence comparison (average training loss) of ResNet20 model on CIFAR-10 dataset for different graph topologies training with (a) 5, (b) 30, and (c) 100 agents.



(a) (b) (c)

Figure 6: Convergence comparison (average training loss) of CNN model for MNIST dataset for different graph topologies training with (a) 5, (b) 30, and (c) 100 agents.

Again since learning rate satisfies $\frac{c_1}{\beta\delta}$, we have:

$$\mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad \mathsf{E}[V(\mathbf{x}^{\tau})] \quad \frac{c_1}{2}\mathsf{E}[k\, r\, V(\mathbf{x}^{\tau})k^2] + \frac{2\sim 1}{2} \tag{37}$$

Now, if we consider the summation of the above inequality for m consecutive iterations, and also based on Lemma 3 we can obtain:

$$V_{\text{inf}} \quad V(\mathbf{x}_1) \quad \mathsf{E}[V(\mathbf{x}^{\tau+1})] \quad V(\mathbf{x}_1) \quad \frac{c_1}{2}\sum_{\tau=1}^{m}\mathsf{E}[k\, r\, V(\mathbf{x}^{\tau})k^2] + \frac{m\ 2\sim 1}{2} \tag{38}$$

Then, by substituting $\sim = \quad + \quad {}^{1}(1 \quad {}_{N}(\quad {}^{0}))$ and rearranging the inequality, we can obtain and prove the theorem inequality. □

A.2 Time for computing MCDS

In this section, we report the time to compute the MCDS for different graph topologies having various number of agents (5, 10, 30, 60, and 100 agents) in Table 3. The values in the Table 3 are the average of three trials.

A.3 Results for ResNet20 model on CIFAR-10 dataset

To evaluate DSMA performance on a complicated model architecture, we trained ResNet20 model on CIFAR-10 dataset. Similar to the trend we had in the main manuscript to present the results, we showed the results for model convergence, accuracy and communication time. Figure 5 shows the model convergence for different graph topologies with 5, 30 and 100 number of agents. Our result on ResNet20 model also compared to different communication-efficient methods in Table 4. Moreover, the scalability of our algorithm is investigated through showing the test accuracy and communication time plots for 60 and 100 agents (Figure 7).

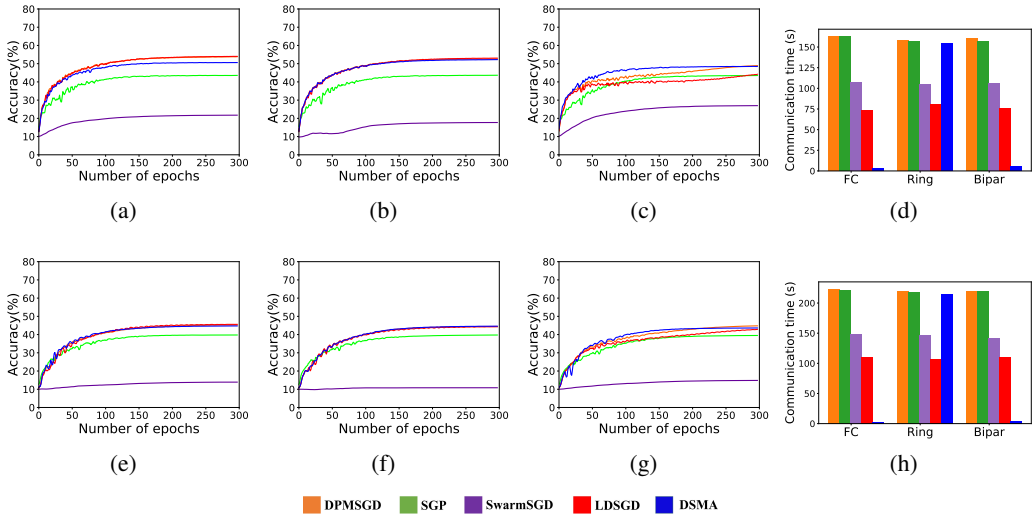


Figure 7: Average test accuracy and communication time (per epoch) of different methods for ResNet20 model on CIFAR-10 data: (a) test accuracy for 60 agents in Fully Connected graph, (b) test accuracy for 60 agents in Ring graph, (c) test accuracy for 60 agents in Bipartite graph, (d) communication time for 60 agents, (e) test accuracy for 100 agents in Fully Connected graph, (f) test accuracy for 100 agents in Ring graph, (g) test accuracy for 100 agents in Bipartite graph, (h) communication time for 100 agents.

Table 4: Test accuracy and communication time for ResNet20 model training on CIFAR10.

			Accuracy (%)					**Communication time (s)**				
	Number of agents		**5**	**10**	**30**	**60**	**100**	**5**	**10**	**30**	**60**	**100**
	DPMSGD	**FC**	70.4	69.4	64.0	53.8	45.3	13.4	28.5	73.8	162.3	223.9
		Ring	70.9	70.2	62.6	52.5	44.5	16.8	26.4	74.6	158.5	220.3
		Bipar	62.4	69.4	62.8	49.6	45.2	15.6	29.3	70.8	160.2	219.7
Communication-efficient methods	**SGP**	**FC**	63.3	55.8	46.7	43.6	39.7	15.2	26.2	71.8	163.4	222.1
		Ring	63.1	55.7	46.7	43.6	39.9	14.7	28.8	72.6	156.8	218.6
		Bipar	63.5	55.5	46.6	43.5	39.5	13.0	28.4	73.6	156.2	219.4
	Matcha	**FC**	55.7	50.3	-	-	-	8.2	16.4	-	-	-
		Ring	55.8	51.7	-	-	-	7.9	14.3	-	-	-
		Bipar	51.5	50.1	-	-	-	7.8	14.1	-	-	-
	LDSGD	**FC**	63.2	**70.0**	**63.9**	**54.0**	**45.6**	9.3	19.0	48.6	107.6	148.4
		Ring	69.5	**70.6**	62.2	53.0	44.3	11.0	17.3	48.4	104.3	146.2
		Bipar	65.7	**70.0**	61.1	44.3	42.9	10.6	17.2	48.1	106.4	142.3
	SwarmSGD	**FC**	**71.4**	60.6	39.2	21.7	13.9	7.2	13.1	35.2	73.2	110.2
		Ring	**70.0**	60.3	37.1	17.7	10.8	**7.8**	**12.6**	**35.2**	**80.1**	**107.3**
		Bipar	**70.2**	60.2	39.9	27.0	14.9	7.7	13.1	35.2	75.2	110.4
	DSMA	**FC**	66.3	65.9	58.3	50.6	44.7	**3.6**	**3.0**	**2.5**	**2.7**	**2.2**
		Ring	66.1	68.6	**62.4**	**53.2**	**44.6**	8.8	22.8	66.8	154.3	215.6
		Bipar	68.3	64.0	**64.6**	**54.5**	**44.6**	**6.1**	**4.9**	**4.6**	**5.3**	**4.4**

A.4 Results for the CNN model on MNIST dataset

To extend our experimental results for a different dataset, we also consider presenting the results for MNIST dataset using the CNN model. The convergence comparison for different graph topologies is presented in Figure 6. Different methods test accuracy and communication time comparison for different number of agents (Table 5) and for large-scale networks (Figure 8) are also investigated.

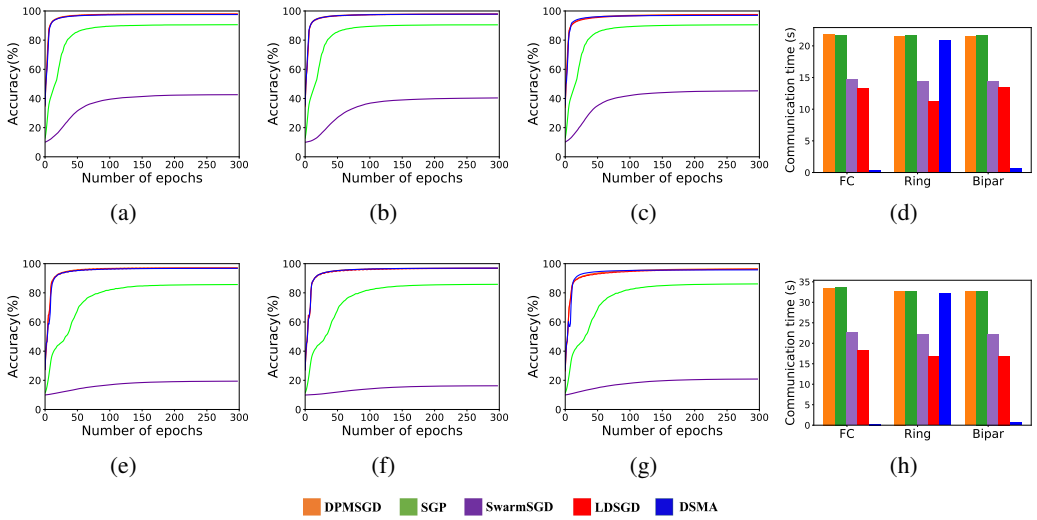


Figure 8: Average test accuracy and communication time (per epoch) of different methods for CNN model on MNIST data: (a) test accuracy for 60 agents in Fully Connected graph, (b) test accuracy for 60 agents in Ring graph, (c) test accuracy for 60 agents in Bipartite graph, (d) communication time for 60 agents, (e) test accuracy for 100 agents in Fully Connected graph, (f) test accuracy for 100 agents in Ring graph, (g) test accuracy for 100 agents in Bipartite graph, (h) communication time for 100 agents.

Table 5: Test accuracy and communication time for CNN model training on MNIST.

			Accuracy (%)					**Communication time (s)**				
	Number of agents		**5**	**10**	**30**	**60**	**100**	**5**	**10**	**30**	**60**	**100**
	DPMSGD	**FC**	98.9	98.9	98.5	97.9	97.1	2.3	4.3	12.4	21.8	33.5
		Ring	98.9	98.9	98.5	97.9	97.0	2.2	4.2	11.7	21.6	32.8
		Bipar	98.9	98.9	98.4	97.6	96.5	2.3	4.3	11.9	21.5	32.7
Communication-efficient methods	**SGP**	**FC**	98.2	97.3	94.0	90.6	85.7	2.3	4.4	12.2	21.7	33.7
		Ring	98.3	97.3	94.0	90.5	85.9	2.5	4.5	12.2	21.7	32.7
		Bipar	98.2	97.3	94.1	90.5	86.2	2.3	4.3	12.0	21.7	32.8
	Matcha	**FC**	98.3	97.9	-	-	-	1.4	2.5	-	-	-
		Ring	98.1	97.6	-	-	-	1.2	2.2	-	-	-
		Bipar	98.1	97.8	-	-	-	1.1	2.1	-	-	-
	LDSGD	**FC**	99.0	98.9	**98.5**	**97.8**	**96.9**	1.5	2.7	8.6	14.7	22.6
		Ring	99.0	98.9	98.4	97.8	96.7	1.6	2.8	8.2	14.4	22.1
		Bipar	99.0	98.9	98.4	97.4	96.7	1.6	2.7	8.4	14.5	22.3
	SwarmSGD	**FC**	98.9	98.2	90.6	42.6	19.4	1.3	2.4	6.7	13.3	18.2
		Ring	98.9	98.1	90.3	40.5	16.3	**1.2**	**2.2**	**7.1**	**11.3**	**16.8**
		Bipar	98.8	98.1	90.8	45.3	20.7	1.3	2.3	6.9	13.5	16.9
	DSMA	**FC**	**99.0**	**98.9**	98.4	97.5	96.8	**0.5**	**0.4**	**0.4**	**0.4**	**0.3**
		Ring	**99.0**	**98.9**	**98.5**	**97.8**	**96.9**	1.5	3.6	11.3	20.9	32.1
		Bipar	**99.0**	**98.9**	**98.4**	**97.8**	**96.9**	**1.0**	**0.8**	**0.8**	**0.7**	**0.6**