A Multi-Token Coordinate Descent Method for Semi-Decentralized Vertical Federated Learning

Pedro Valdeira*† CMU, IST
Yuejie Chi* CMU
Cláudia Soares‡ NOVA
João Xavier† IST

September 19, 2023

Abstract

Communication efficiency is a major challenge in federated learning (FL). In client-server schemes, the server constitutes a bottleneck, and while decentralized setups spread communications, they do not necessarily reduce them due to slower convergence. We propose Multi-Token Coordinate Descent (MTCD), a communication-efficient algorithm for semi-decentralized vertical federated learning, exploiting both client-server and client-client communications when each client holds a small subset of features. Our multi-token method can be seen as a parallel Markov chain (block) coordinate descent algorithm and it subsumes the client-server and decentralized setups as special cases. We obtain a convergence rate of $\mathcal{O}(1/T)$ for nonconvex objectives when tokens roam over disjoint subsets of clients and for convex objectives when they roam over possibly overlapping subsets. Numerical results show that MTCD improves the state-of-the-art communication efficiency and allows for a tunable amount of parallel communications.

1 Introduction

Federated Learning (FL) is a machine learning paradigm where data is distributed across a set of clients who collaborate to learn a model without sharing local data (McMahan et al., 2017). Most FL literature considers data distributed by samples (horizontal FL), where each client holds all the features of a subset of the samples, yet recently there has been a growing interest on feature-distributed setups (vertical FL), where each client holds a subset of the features for all samples (He et al., 2018; Chen et al., 2020; Alghunaim et al., 2021; Liu et al., 2022b; Castiglia et al., 2022). Vertical FL may be of particular interest for applications using, for example, time series data measured by personal devices to learn a model of some cross-client phenomenon of interest (e.g. meteorological), where each sample corresponds to the data collected across the devices at a given timestamp. Another example involves performing a computer vision task using multiple views of the same object, where a sample corresponds to the concatenation of views from different devices.

FL often deals with the client-server setup, where a server is connected to all clients and the clients do not communicate with each other, forming a star-shaped topology. However, such schemes have a single point of failure and suffer from a communication bottleneck on the server (Lian et al., 2017). On the other hand, there is extensive literature on decentralized optimization, where there is no server—from earlier work motivated by applications such as wireless sensor networks and multiagent control (Nedic and Ozdaglar, 2009; Duchi et al., 2012; Qu and Li, 2018), to recent work motivated by FL (Koloskova et al., 2020; Li et al., 2020; Zhao et al., 2022). Yet, these algorithms often converge slowly in sparse and large networks (Nedic et al., 2018) and, although they spread the communication load across the network, they tend to have poor communication efficiency. Semi-Decentralized FL (SDFL) uses both client-server *and* client-client communications, reducing the overhead at the server (Lin et al., 2021) while avoiding the shortcomings of decentralized setups and being able to handle multiple clusters of clients.

*Department of Electrical and Computer Engineering, Carnegie Mellon University; email: pvaldeira@cmu.edu.
†Institute for Systems and Robotics, Instituto Superior Técnico.
‡Department of Computer Science, NOVA School of Science and Technology.
§A preliminary version of this work was presented at the 2022 NeurIPS Workshop on Federated Learning: Recent Advances and New Challenges (Valdeira et al., 2022).

When concerned with the communications between clients, the use of a token method (Bertsekas, 1997; Nedic and Bertsekas, 2001; Ram et al., 2009; Johansson et al., 2010; Mao et al., 2020; Hendrikx, 2022), where a model-embedding token follows a random walk over a communication graph (undergoing local updates), allows for better communication efficiency (Hendrikx, 2022) than the more common consensus-based methods (Nedic and Ozdaglar, 2009; Duchi et al., 2012; Qu and Li, 2018; Koloskova et al., 2020). Yet, the convergence rate of token methods degrades even faster for larger and sparser networks, due to a lack of parallel communications (Hendrikx, 2022). Based on the idea that performing multiple random walks in parallel leads to a linear speed-up in the cover time (Alon et al., 2008), multi-token methods (Ye et al., 2020; Chen et al., 2022; Hendrikx, 2022) mitigate this problem by running multiple tokens simultaneously and combining them.

1.1 Our contribution

Motivated by the above observations, we propose an SDFL multi-token algorithm for vertical FL. Our main contributions are as follows.

- We introduce Multi-Token Coordinate Descent (MTCD), which, to the best of our knowledge, is the first multi-token method leveraging the SDFL scheme to achieve a flexible degree of dependence on the server, recovering both client-server and fully decentralized FL as special cases;
- We show that MTCD converges at a rate of $\mathcal{O}(1/T)$ for nonconvex objectives when tokens roam over disjoint subsets of clients and for convex objectives when they roam over possibly overlapping subsets of clients;
- Numerical experiments on both synthetic and real data and for a variety of communication setups show that MTCD improves the state-of-the-art communication efficiency.

1.2 Related works

Coordinate descent. Coordinate Descent (CD) methods (Wright, 2015), where (blocks of) coordinates are updated sequentially, rather than simultaneously, are natural candidates for optimization in feature-distributed learning. The block selection is most often cyclic (Beck and Tetruashvili, 2013) or independent and identically distributed at random (Nesterov, 2012; Richtárik and Takáč, 2012). In contrast, Sun et al. (2019) considers block selection following a Markov chain. Several extensions to CD have been proposed, such as acceleration and parallelization (Fercoq and Richtárik, 2015) and distributed CD methods (Liu et al., 2022b; Chen et al., 2022).

Vertical FL. Existing vertical FL works include He et al. (2018), which generalizes Smith et al. (2018) to the decentralized setting. Both works use primal-dual optimization techniques, as does DCPA (Alghunaim et al., 2021), a state-of-the-art decentralized method. In contrast, the following methods work in the primal domain, allowing them to learn more general models. In (Chen et al., 2020), a CD-based method is used, but no local updates are performed, while Liu et al. (2022b) does consider local updates. This latter work and Castiglia et al. (2022), which introduces the use of compressed embeddings, lowering the communication cost, are particularly close to our method. Note that the communication costs in vertical FL methods typically depend on the number of samples (or batch size, in the case of stochastic methods) (Alghunaim et al., 2021; Liu et al., 2022b; Castiglia et al., 2022), further highlighting the importance of communication efficiency. An interesting line of work related to vertical FL is hybrid FL (Zhang et al., 2021b), which deals with datasets distributed both by samples and features. For a more detailed survey of vertical FL methods, see (Wei et al., 2022; Liu et al., 2022a).

Semi-Decentralized FL. Recently, SDFL approaches have been proposed to lower communication costs and deal with data heterogeneity (Lin et al., 2021; Guo et al., 2021), and to handle intermittent connections, latency, and stragglers (Bian and Xu, 2022; Yemini et al., 2022). Additionally, other SDFL works deal with (multi-layered) hierarchical networks (Zhang et al., 2021a; Hosseinalipour et al., 2022). SDFL is also referred to as hybrid FL sometimes, but we opt for the term semi-decentralized FL to avoid confusion with the data partitioning setting mentioned above.

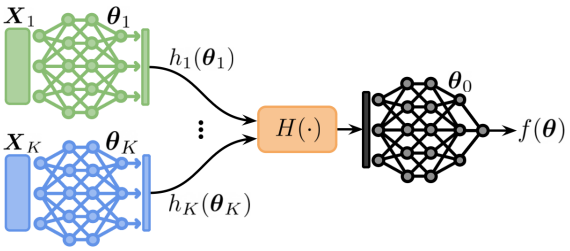


Figure 1: A split neural network, where K embeddings are obtained from neural networks, before an aggregation mechanism H is applied and its result is inputted into a fusion neural network.

2 Problem setup

We consider a dataset $\boldsymbol{X} \in \mathbb{R}^{N \times d}$ with N d-dimensional samples distributed by features across a set of clients $[K] \coloneqq \{1, \dots, K\}$. Client k holds its local data $\boldsymbol{X}_k \in \mathbb{R}^{N \times d_k}$ and we have $\boldsymbol{X} = [\boldsymbol{X}_1, \cdots, \boldsymbol{X}_K]$. Note that $d_1 + \cdots + d_K = d$. We consider a broad class of machine learning models, know as split neural networks (Ceballos et al., 2020), illustrated in Figure 1.

In split neural networks, each client has an associated local model $h_k(\cdot\,; \boldsymbol{X}_k) \colon \boldsymbol{\Theta}_k \mapsto \mathcal{H}_k$ parameterized by $\boldsymbol{\theta}_k \in \boldsymbol{\Theta}_k$, which extracts a (typically lower-dimensional) representation of $\boldsymbol{X}_k$. For simplicity, we write $h_k(\boldsymbol{\theta}_k) \coloneqq h_k(\boldsymbol{\theta}_k; \boldsymbol{X}_k)$. These embeddings, $h_k(\boldsymbol{\theta}_k)$, are then combined using an aggregation mechanism $H \colon \mathcal{H}_1 \times \cdots \times \mathcal{H}_K \mapsto \mathcal{H}$ to form $H(h_1(\boldsymbol{\theta}_1), \dots, h_K(\boldsymbol{\theta}_K))$, which is used as input to a fusion model $\phi \colon \mathcal{H} \times \boldsymbol{\Theta}_0 \mapsto \mathbb{R}$, parameterized by $\boldsymbol{\theta}_0$. Although more aggregation mechanisms are possible (Ceballos et al., 2020), we focus on aggregation by concatenation (the most general case) and by sum. While $\boldsymbol{\theta}_k \in \boldsymbol{\Theta}_k$ is associated with $\boldsymbol{X}_k$, acting as a local model for client k, our fusion model $\boldsymbol{\theta}_0$ can be learned at different locations, depending on whether we consider the existence of a server. Split neural networks include, for example, generalized linear models, such as linear regression, logistic regression, and support vector machines, where $h_k(\boldsymbol{\theta}_k) = \boldsymbol{X}_k \boldsymbol{\theta}_k$ for $k \in [K]$ and $\boldsymbol{\Theta}_0$ is an empty set.

Let $\boldsymbol{\Theta}$ denote a parameter space such that $\boldsymbol{\Theta} = \boldsymbol{\Theta}_0 \times \boldsymbol{\Theta}_1 \times \cdots \times \boldsymbol{\Theta}_K$ and $f \colon \boldsymbol{\Theta} \mapsto \mathbb{R}$ denote our objective function, our goal is to solve the following optimization problem, which encompasses the training of split neural networks:

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) \coloneqq \phi\left(H\left(h_1(\boldsymbol{\theta}_1; \boldsymbol{X}_1), \dots, h_K(\boldsymbol{\theta}_K; \boldsymbol{X}_K)\right), \boldsymbol{\theta}_0\right) \right\}, \tag{1}$$

where the labels $\boldsymbol{y}$ are included in the loss function, which we assume to be known by all clients.

Throughout the paper, we consider Problem (1) and assume f is an L-smooth function. The standard definition of L-smoothness is given below. We define and assume $f^\star \coloneqq \min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) > -\infty$.

Assumption 1 (Smoothness)**.** *A differentiable function $f \colon \mathbb{R}^d \mapsto \mathbb{R}$ is L-smooth if there exists some $L \in (0, \infty)$:*

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d. \tag{A1}$$

3 Proposed method

3.1 The fully decentralized setting

We start by introducing a simple, special case of our algorithm, which we refer to as Single-Token Coordinate Descent (STCD). This method, which is also a subroutine of our general algorithm, is closely related to Sun et al. (2019) and the application mentioned therein, taken from Mao et al. (2020). Yet, we work in the primal domain and on a feature-distributed setting.

Setup. In this section, we do not require the existence of a server. We solve Problem (1) in a fully decentralized manner, communicating through channels described by a communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \coloneqq [K]$ is the vertex set and $\mathcal{E}$ the edge set. We denote the set of neighbors of client k by

Algorithm 1: Single-Token Coordinate Descent

Input : initial point $\boldsymbol{\theta}^{0,0}$, initial client k^0, step-size η, number of hops S, number of local updates Q

$\mathcal{Z}^{0,0} \leftarrow \{H(h_1(\boldsymbol{\theta}_1^{0,0}; \boldsymbol{X}_1), \ldots, h_K(\boldsymbol{\theta}_K^{0,0}; \boldsymbol{X}_K)), \boldsymbol{\theta}_0^{0,0}\}$

$\boldsymbol{\theta}^{S,Q}, \mathcal{Z}^{S,Q}, k^S \leftarrow$ `TokenRoaming(`$\boldsymbol{\theta}^{0,0}, \mathcal{Z}^{0,0}, k^0, \boldsymbol{X}, S, Q$`)`

Function `TokenRoaming(`$\boldsymbol{\theta}, \mathcal{Z}, k, \boldsymbol{D}, S, Q$`):`

  $\boldsymbol{\theta}^{0,0}, \mathcal{Z}^{0,0}, k^0 \leftarrow \boldsymbol{\theta}, \mathcal{Z}, k$

  for $s = 0, \ldots, S$ **do**

    for $q = 0, \ldots, Q-1$ **in client** k^s **do**

      Compute $\boldsymbol{\theta}_{k^s}^{s,q+1}$ via a CD step (2), using $\mathcal{Z}^{s,q}$, $\boldsymbol{\theta}_{k^s}^{s,q}$, and $\boldsymbol{D}_{k^s}$

      Compute $\mathcal{Z}^{s,q+1}$ from $\mathcal{Z}^{s,q}$, $\boldsymbol{\theta}_{k^s}^{s,q}$, $\boldsymbol{\theta}_{k^s}^{s,q+1}$, and $\boldsymbol{D}_{k^s}$

    Client k^s sends $\mathcal{Z}^{s,Q}$ to client $k^{s+1} \sim \mathcal{U}(\bar{\mathcal{N}}_{k^s})$

    $\boldsymbol{\theta}^{s+1,0}, \mathcal{Z}^{s+1,0} \leftarrow \boldsymbol{\theta}^{s,Q}, \mathcal{Z}^{s,Q}$

  return $\boldsymbol{\theta}^{S,Q}, \mathcal{Z}^{S,Q}, k^S$

$\mathcal{N}_k \coloneqq \{i \colon \{i,k\} \in \mathcal{E}\}$ and define $\bar{\mathcal{N}}_k \coloneqq \mathcal{N}_k \cup \{k\}$. In this section only, $\boldsymbol{\theta}_0$ is associated with some client k, which is responsible for updating both the local model it holds, $\boldsymbol{\theta}_k$, and the fusion model $\boldsymbol{\theta}_0$.[1]

Token. Since all clients know ϕ, if a client knows $\mathcal{Z} \coloneqq \{H(h_1(\boldsymbol{\theta}_1), \ldots, h_K(\boldsymbol{\theta}_K)), \boldsymbol{\theta}_0\}$, it can compute f. We call $\mathcal{Z}$ our *token*. The size of the token depends on the model being considered. For example, if $\mathcal{H}_k \subseteq \mathbb{R}^{NE}$ for all k (i.e., we have an E-dimensional embedding per sample) and we do aggregation by concatenation, then $\mathcal{Z}$ is of size $KNE + \dim(\boldsymbol{\Theta}_0)$, where $\dim(\cdot)$ denotes the dimensionality of a space. Yet, for aggregation by sum, we drop the dependency on K. In particular, for generalized linear models, $\mathcal{Z} = \{\boldsymbol{X}\boldsymbol{\theta}\}$ is of size N.

The token suffices to perform local CD steps. We have seen that a client holding $\mathcal{Z}$ can compute f. Yet, more importantly, if client k has access to its local data $\boldsymbol{X}_k$ and local model $\boldsymbol{\theta}_k$, then holding $\mathcal{Z}$ enables it to compute the partial gradient with respect to $\boldsymbol{\theta}_k$,

$$\begin{aligned}\nabla_k f(\boldsymbol{\theta}) \coloneqq \nabla_{\boldsymbol{\theta}_k} f(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}_k} \phi\left(H(h_1(\boldsymbol{\theta}_1), \ldots, h_K(\boldsymbol{\theta}_K)), \boldsymbol{\theta}_0\right) \\ &= \frac{d\phi\left(H(h_1(\boldsymbol{\theta}_1), \ldots, h_K(\boldsymbol{\theta}_K)), \boldsymbol{\theta}_0\right)}{dH(h_1(\boldsymbol{\theta}_1), \ldots, h_K(\boldsymbol{\theta}_K))} \cdot \frac{dH(h_1(\boldsymbol{\theta}_1), \ldots, h_K(\boldsymbol{\theta}_K))}{dh_k(\boldsymbol{\theta}_k)} \cdot \frac{dh_k(\boldsymbol{\theta}_k)}{d\boldsymbol{\theta}_k},\end{aligned}$$

where $\mathcal{Z}$ is used in the computation of the first two terms. This will allow client k to update its local model $\boldsymbol{\theta}_k$.

We now describe STCD, summarized in Algorithm 1, where $\mathcal{U}$ denotes the uniform distribution. We index $\boldsymbol{\theta}$ and $\mathcal{Z}$ with two counters: one for the sequence of clients (and thus coordinate block) visited while roaming and one for the local updates at each client. To simplify the description of the algorithm, we omit $\boldsymbol{\theta}_0$ for the rest of Section 3.1, as if it were part of the local model of the client it is associated with for updating purposes. Yet, $\boldsymbol{\theta}_k$ does not leave k but $\boldsymbol{\theta}_0$ does, as $\boldsymbol{\theta}_0$ is part of the token.

- **Initialization:** The token $\mathcal{Z}^{s,q}$ must always be in accordance with $\boldsymbol{\theta}^{s,q}$. So, as $\mathcal{Z}^{0,0}$ starts at client k^0, this client must know $\{H(h_1(\boldsymbol{\theta}_1^{0,0}), \ldots, h_K(\boldsymbol{\theta}_K^{0,0})), \boldsymbol{\theta}_0^{0,0}\}$. For some models, we can achieve this by initializing the local models $\boldsymbol{\theta}_k^{0,0}$ such that the embeddings $h_k(\boldsymbol{\theta}_k^{0,0})$ are independent of local data $\boldsymbol{X}_k$. When this is not possible, the clients can send their initial embeddings to k^0 as a prelude.

- **Updating the local model and the token:** as explained above, the client holding the token after s hops, k^s, can compute the partial gradient with respect to its local model $\boldsymbol{\theta}_{k^s}$ locally. This allows it to perform a CD step. Further, to lower communication costs, we do Q local CD updates at each client. That is, for $q = 0, \ldots, Q-1$:

$$\boldsymbol{\theta}_{k^s}^{s,q+1} = \boldsymbol{\theta}_{k^s}^{s,q} - \eta \nabla_{k^s} f(\boldsymbol{\theta}^{s,q}) \tag{2}$$

[1] We do this for alignment with the analysis in Sun et al. (2019), where all blocks are selected following a Markov Chain. However, in practice, we may want to update $\boldsymbol{\theta}_0$ at each client instead, in which case the analysis would need to be adjusted.

and $\boldsymbol{\theta}_k^{s,q+1} = \boldsymbol{\theta}_k^{s,q}$ for $k \neq k^s$. We must now update $\mathcal{Z}$ accordingly. To compute $\mathcal{Z}^{s,q+1}$, we use $\mathcal{Z}^{s,q}$, $h_{k^s}(\boldsymbol{\theta}_{k^s}^{s,q+1})$, and $h_{k^s}(\boldsymbol{\theta}_{k^s}^{s,q})$, which are held by k^s. For example, for aggregation by sum, we have

$$H(h_1(\boldsymbol{\theta}_1^{s,q+1}), \ldots, h_K(\boldsymbol{\theta}_K^{s,q+1})) = H(h_1(\boldsymbol{\theta}_1^{s,q}), \ldots, h_K(\boldsymbol{\theta}_K^{s,q})) + h_{k^s}(\boldsymbol{\theta}_{k^s}^{s,q+1}) - h_{k^s}(\boldsymbol{\theta}_{k^s}^{s,q}).$$

This allows us to perform multiple local CD steps. Further, after these Q steps, the updated token can be sent to client k^{s+1}. Thus, by induction, we keep the token up-to-date throughout our algorithm.

- **Communicating the token:** the token is communicated to a neighbor of k^s. This results in a sequence of clients (and blocks) that follows a Markov Chain.

In essence, STCD is a technique allowing for Markov Chain Coordinate Descent (Sun et al., 2019) to be performed in feature-distributed setups. In terms of the progress made in the parameter space, Algorithm 1 differs from Sun et al. (2019) only in that we consider local updates with $Q > 1$.

Convergence guarantees. If f is an L-smooth function (A1) with a nonempty set of minimizers and $(\boldsymbol{\theta}^i)_{i=1}^r$ is a sequence generated by Algorithm 1, Sun et al. (2019) give convergence guarantees for $Q = 1$. In particular, let $r = sQ + q$, we have under mild assumptions on the Markov chain (for example, if the Markov chain is time-homogeneous, irreducible, and aperiodic) that $\lim_{r\to\infty} \mathbb{E}\|\nabla f(\boldsymbol{\theta}^r)\| = 0$ and, let $\Delta := f\left(\boldsymbol{\theta}^0\right) - f^\star$,

$$\mathbb{E}\left[\min_{i\in[r]} \|\nabla f(\boldsymbol{\theta}^i)\|^2\right] \leq \frac{(\Omega_1(\tau-1)^2 + \Omega_2)\Delta}{r}, \tag{3}$$

where Ω_1 and Ω_2 are constants that depend on the minimum value of the stationary distribution of the Markov chain $\pi_{\min}$, the step-size η, and the smoothness constant L. Further, τ denotes the $\frac{\pi_{\min}}{2}$-mixing time of the the Markov chain. (We present all the aforementioned Markov chain-related terms in Appendix A.) While Sun et al. (2019) only consider $Q = 1$ explicitly, their analysis can also cover the $Q > 1$ case. To see this, consider a virtual dynamic graph $\mathcal{G}^r = (\mathcal{V}, \mathcal{E}^r)$ where $\mathcal{V}$ is the original vertex set and, recalling that $\mathcal{E}$ is the original edge set,

$$\mathcal{E}^r = \begin{cases} \mathcal{E} & \text{if } r \bmod Q = 0, \\ \{\{i,i\} : i \in \mathcal{V}\} & \text{otherwise.} \end{cases}$$

Running STCD on $\mathcal{G}^r$ with a single local update is equivalent to running STCD on the original graph $\mathcal{G}$ with Q local updates. This dynamic graph preserves the properties required for the analysis to hold. To see this, let $\boldsymbol{P}$ denote the transition matrix of a random walk on the original graph $\mathcal{G}$ and let $\boldsymbol{P}(r)$ denote the transition matrix of a random walk on $\mathcal{G}^r$. Note that $\boldsymbol{P}(r) = \boldsymbol{I}$ for all $r \mod Q \neq 0$, where $\boldsymbol{I}$ denotes the identity matrix, and $\boldsymbol{P}(r) = \boldsymbol{P}$ for all $r \mod Q = 0$. Assuming, for simplicity, that $R \geq Q$, we have that $\boldsymbol{P}(r)\boldsymbol{P}(r+1)\ldots\boldsymbol{P}(r+R) = \boldsymbol{P}^{\lfloor R/Q \rfloor}$. We thus recover the results in Sun et al. (2019) up to a factor of Q in the mixing time. (That is, in (3), we replace τ by $Q\tau$.)

Limitations. The decentralized token algorithm in this section has an appealing simplicity. However, while it outperforms state-of-the-art feature-distributed learning algorithms in a variety of setups, as we will see in Section 4, its performance deteriorates faster with network connectivity than these decentralized consensus-based algorithms and its convergence per iteration can be rather slow. These problems will be mitigated by the more general multi-token method presented next.

3.2 Semi-decentralized setting

In Section 3.1, we introduced a special case of MTCD where a single token roams over a fully decentralized set of clients. We now present our method in the semi-decentralized setting, which subsumes the setting in Section 3.1 as a special case. Multi-token CD alternates between a *roaming* step and a *syncing* step.

- *Roaming.* We start with multiple, matching tokens at a subset of the clients. As each token performs a different random walk realization, they undergo different sequences of CD updates, becoming distinct.
- *Syncing.* To leverage these parallel computations while keeping our model estimates coupled, we periodically sync the roaming tokens at the server, combining the progress of multiple CD sequences.

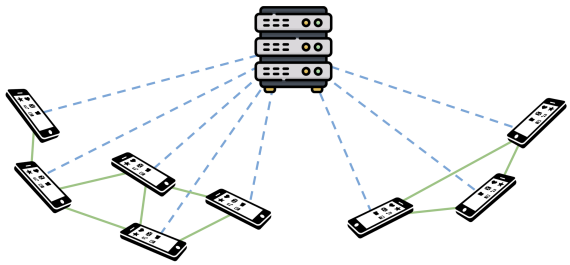


Figure 2: Semi-decentralized setup. Client-server communications are represented by dashed blue lines and client-client communications by solid green lines.

By alternating between these two steps, we get a communication-efficient algorithm with a flexible degree of parallelization, depending on the number of tokens and the frequency at which we sync them. This allows us to smoothly control the trade off between the communication efficiency of settings with less parallel computations and the faster iteration convergence of settings with more parallel computations.

We further generalize STCD by allowing the use of stochastic gradient estimates. Let $\boldsymbol{B} \in \mathbb{R}^{B \times d}$ denote a mini-batch of size B defined by a random set of indices $\mathcal{B} \subseteq [N]$, we make the following standard assumptions that our gradient estimate is unbiased and has a bounded variance. For clarity, we make the dependency on $\boldsymbol{X}$ explicit by writing $\nabla f(\boldsymbol{\theta}; \boldsymbol{X}) = \nabla f(\boldsymbol{\theta})$.

Assumption 2 (Unbiased gradients)**.** *For any mini-batch $\boldsymbol{B}$, the stochastic gradient is unbiased:*

$$\mathbb{E}_{\mathcal{B}}[\nabla f(\boldsymbol{\theta}; \boldsymbol{B})] = \nabla f(\boldsymbol{\theta}; \boldsymbol{X}), \quad \forall\, \boldsymbol{\theta} \in \mathbb{R}^d. \tag{A2}$$

Assumption 3 (Bounded variance)**.** *For any mini-batch $\boldsymbol{B}$, there exists a constant $\sigma \geq 0$ such that:*

$$\mathbb{E}_{\mathcal{B}} \|\nabla f(\boldsymbol{\theta}; \boldsymbol{B}) - \nabla f(\boldsymbol{\theta}; \boldsymbol{X})\|^2 \leq \frac{\sigma^2}{B}, \quad \forall\, \boldsymbol{\theta} \in \mathbb{R}^d. \tag{A3}$$

Setup. In addition to a communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we now also consider the existence of a central server with links to all clients, as illustrated in Figure 2. The existence of a server brings a change to the model partitioning: $\boldsymbol{\theta}_0$ is now updated at the server. Further, we now have Γ tokens roaming simultaneously. Each token $\mathcal{Z}_\gamma$ has an associated model estimate $\boldsymbol{\theta}(\gamma)$, for $\gamma \in [\Gamma]$. Thus, during the roaming step, each client k must now keep up to Γ local model estimates $\boldsymbol{\theta}_k(\gamma)$ in memory.[2] To simplify the description of the algorithm, we define $[\Gamma]_0 \coloneqq [\Gamma] \cup \{0\}$, with token $\gamma = 0$ staying at the server throughout the roaming step.

We now describe MTCD, summarized in Algorithm 2, where $k_\gamma^{t,s}$ denotes the client holding token γ after t synchronizations and s hops and P_γ denotes a distribution over the clients, for $\gamma \in [\Gamma]$. We index $\boldsymbol{\theta}$ and $\mathcal{Z}$ with three counters: one for synchronizations at the server and two matching the counters used in STCD. For simplicity, we write $\mathcal{Z}^t \coloneqq \mathcal{Z}_1^{t,0,0} = \cdots = \mathcal{Z}_\Gamma^{t,0,0}$ and $\boldsymbol{\theta}^t \coloneqq \boldsymbol{\theta}^{t,0,0}(1) = \cdots = \boldsymbol{\theta}^{t,0,0}(\Gamma)$.

- **Initialization:** all model-token pairs $(\boldsymbol{\theta}(\gamma), \mathcal{Z}_\gamma)$ are initialized to the same values. As in STCD, $\mathcal{Z}_\gamma$ must be in accordance with $\boldsymbol{\theta}(\gamma)$.
- **Roaming:** The server samples a set of indices $\mathcal{B}^t$ and communicates it to each client k, which returns its local embedding $h_k(\boldsymbol{\theta}_k^t; \boldsymbol{B}^t)$. This allows the server to compute token $\mathcal{Z}^t$ and send copies of it to start the roaming step at $k_\gamma^{t,0} \sim P_\gamma$, where $P_\gamma(k) \coloneqq \mathbb{P}(k_\gamma^{t,0} = k)$. Note that P_0 is a point mass distribution with support over the server, $k = 0$ (a node without neighbors). As $\mathcal{Z}_\gamma$ reaches client $k_\gamma^{t,s}$, it is used to perform a local CD step on model estimate $\boldsymbol{\theta}(\gamma)$ with respect to block $k_\gamma^{t,s}$ and is then updated accordingly. Each $\mathcal{Z}_\gamma$ roams for S hops, as in STCD. In parallel, $\boldsymbol{\theta}_0(0)$ is updated at the server.
- **Syncing:** after S hops, each client combines its model estimates, obtaining $\boldsymbol{\theta}_k^{t+1} = \sum_{\gamma=1}^{\Gamma} w_{k\gamma} \boldsymbol{\theta}_k^{t,S,Q}(\gamma)$. We cover the choice of $\boldsymbol{w}_k \coloneqq (w_{k1}, \ldots, w_{k\Gamma})$, which lies in the Γ-dimensional probability simplex, later.

[2] In practice, it suffices to to add a copy of $\boldsymbol{\theta}^t$ as a new token visits during a given roaming step (with a maximum of Γ model estimates at a client), resetting the number of copies when syncing.

Algorithm 2: Multi-Token Coordinate Descent

Input : initial point $\boldsymbol{\theta}^0$, step-size η, number of hops S, number of local updates Q, model estimates combination weights $\{w_{k\gamma}\}$, distributions $\{P_\gamma\}$

for $t = 0, \dots, T-1$ **do**
  Server samples batch indices $\mathcal{B}^t \subseteq [N]$ and sends them to clients
  for $k \in [K]$ **in parallel do**
    Client k sends its local embedding $h_k(\boldsymbol{\theta}_k^t; \boldsymbol{B}^t)$ to the server
  Server computes token $\mathcal{Z}^t \leftarrow \{H(h_1(\boldsymbol{\theta}_1^t; \boldsymbol{B}^t), \dots, h_K(\boldsymbol{\theta}_K^t; \boldsymbol{B}^t)), \boldsymbol{\theta}_0^t\}$
  for $\gamma \in [\Gamma]_0$ **in parallel do**
    Server sends $\mathcal{Z}^t$ to $k_\gamma^{t,0} \sim P_\gamma$
    $\boldsymbol{\theta}^{t,S,Q}(\gamma), \mathcal{Z}_\gamma^{t,S,Q}, k_\gamma^{t,S} \leftarrow$ `TokenRoaming`$(\boldsymbol{\theta}^t, \mathcal{Z}^t, k_\gamma^{t,0}, \boldsymbol{B}^t, S, Q)$
  for $k \in [K]_0$ **in parallel do**
    $\boldsymbol{\theta}_k^{t+1} = \sum_{\gamma=1}^{\Gamma} w_{k\gamma} \boldsymbol{\theta}_k^{t,S,Q}(\gamma)$

Recovering client-server and decentralized setups. On the one hand, if no client-server communications are available ($S \to \infty$), our algorithm is reduced to a fully decentralized one, recovering STCD. On the other hand, if the edge set $\mathcal{E}$ is empty, we recover the client-server setting. In this case, if we assign a token to each client ($\Gamma = K$ and each P_γ has support over a single different client), we get full participation, as in Liu et al. (2022b). In contrast, if $\Gamma < K$, we recover a partial participation client-server scheme. To the best of our knowledge, this is the first partial participation vertical FL algorithm with multiple local updates.

In general, as we increase the amount of parallel computations (by increasing Γ) and client-server communications (by lowering S), the communication efficiency and the number of iterations needed to converge both decrease. Given this trade-off, we see that our choice of S and Γ depends on the application.

Having introduced the general MTCD algorithm, we now go over two particular instances of it, both for semi-decentralized setups, and present convergence guarantees for each.

Setting with a token per cluster. We now present some convergence guarantees for MTCD in the case where we have C disjoint clusters of clients $\mathcal{C}_1, \dots, \mathcal{C}_C$ and one token roams each cluster, hence $\Gamma = C$. Without loss of generality, we assume γ roams $\mathcal{C}_\gamma$, allowing us to use γ and c interchangeably. Thus, P_γ has support over $\mathcal{C}_\gamma$, and only over $\mathcal{C}_\gamma$. We also define a cluster containing only the server, $\mathcal{C}_0 \coloneqq \{0\}$.

These clusters may correspond to the natural topology of the communication graph, due to physical limitations or to privacy constraints preventing communications between clients in different clusters (e.g., households or companies). Yet, we may also generate artificial partitioning of the original graph, prior to learning our model, in order to allow for the use of multiple tokens while avoiding overlapping trajectories, rather than having a single token roaming over a too large (and thus poorly connected) graph.

Since in this setting the blocks of coordinates being updated in each model estimate are disjoint, we let $\boldsymbol{w}_k \in \mathbb{R}^\Gamma$ be the one-hot encoding for the token that visits client k, thus combining the model estimates by simply taking the updated version of each block.

Theorem 1. *Let f have a nonempty set of minimizers and let all $f(\cdot; \boldsymbol{B})$ be L-smooth (A1) and have an unbiased gradient (A2) with a bounded variance (A3). If $(\boldsymbol{\theta}^t)$ is a sequence generated by Algorithm 2 with $\eta \in \left(0, \frac{\rho}{(L+1)SQ(\rho+Se(1+e))}\right)$ under the token per cluster setting (that is, P, $\mathcal{G}$, and $\boldsymbol{w}_k$ as explained above) and $\mathbb{P}(k_\gamma^{t,0} = k) > 0$ for all γ and $k \in \mathcal{C}_\gamma$, then:*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \left\|\nabla f\left(\boldsymbol{\theta}^t\right)\right\|^2\right] = \mathcal{O}\left(\frac{\Delta}{T} + \frac{\sigma^2}{B}\right), \tag{4}$$

where the expectation is over $\{k_c^{t,s}\}$ and $\{\mathcal{B}_t\}$. Here, $\Delta \coloneqq f\left(\boldsymbol{\theta}^0\right) - f^\star$.

Encouragingly, we see that, for full batch (exact gradient), we recover the $\mathcal{O}(\Delta/T)$ rate of convergence for the (expected) squared norm of the gradient of centralized CD methods. Further, for mini-batches,

by choosing a sufficiently large batch size $B = \Omega(\sigma^2/\epsilon)$ we can preserve the iteration complexity to reach $\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\left\|\nabla f\left(\boldsymbol{\theta}^t\right)\right\|^2\right] \leq \epsilon$.

Setting with overlapping token trajectories. We now present some convergence guarantees for MTCD in the setting where we allow for overlapping token trajectories. We propose choosing the convex combination weights to be $\boldsymbol{w}_k = (\frac{1}{\Gamma}, \ldots, \frac{1}{\Gamma}) \in \mathbb{R}^\Gamma$ for all k, thus combining the model estimates by averaging them. We also consider, for simplicity, that $P_\gamma = P$, and assume that distribution P has support over all clients.

In this setting, to handle the periodic combination by averaging of the model estimates, we develop convergence guarantees for convex objectives. The standard definition of convexity is given below.

Assumption 4 (Convexity)**.** *A function $f\colon \mathbb{R}^d \mapsto \mathbb{R}$ is convex if for all $a \in [0,1]$:*

$$f(a\boldsymbol{x} + (1-a)\boldsymbol{y}) \leq af(\boldsymbol{x}) + (1-a)f(\boldsymbol{y}), \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d. \tag{A4}$$

Theorem 2. *Let f be convex (A4) and have a nonempty set of minimizers and let all $f(\cdot;\boldsymbol{B})$ be L-smooth (A1) and have an unbiased gradient (A2) with a bounded variance (A3). If $(\boldsymbol{\theta}^t)$ is a sequence generated by Algorithm 2 with for $\eta \in \left(0, \frac{\rho'}{(L+1)SQ(\rho'+Se(1+e))}\right)$ under the overlapping tokens setting (that is, P, $\mathcal{G}$, and $\boldsymbol{w}_k$ as explained above) and $\mathbb{P}(k_\gamma^{t,0} = k) > 0$ for all $\gamma \in [\Gamma]$ and $k \in [K]$, then:*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\left\|\nabla f\left(\boldsymbol{\theta}^t\right)\right\|^2\right] = \mathcal{O}\left(\frac{\Delta}{T} + \frac{\sigma^2}{B}\right),$$

where the expectation is over $\{k_\gamma^{t,s}\}$ and $\{\mathcal{B}_t\}$. Here, $\Delta \coloneqq f\left(\boldsymbol{\theta}^0\right) - f^\star$.

As in the token per cluster setting, we recover the $\mathcal{O}(\Delta/T)$ rate for the (expected) squared norm of the gradient when the exact gradient is used, matching the rate of centralized CD methods, and, for a sufficiently large mini-batch size $B = \Omega(\sigma^2/\epsilon)$, we preserve the iteration complexity to reach $\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\left\|\nabla f\left(\boldsymbol{\theta}^t\right)\right\|^2\right] \leq \epsilon$.

4 Experiments

We test our method empirically, comparing it to DCPA (Alghunaim et al., 2021), a state-of-the-art fully decentralized method, and a standard vertical FL (S-VFL) method which, since we do not consider compression in our experiments, coincides with both C-VFL (Castiglia et al., 2022) and FedBCD (Liu et al., 2022b). Note that, while some trajectories have a small variance, making the confidence interval hard to see, all experiments are run for 5 seeds.

4.1 Convex problems

In this section, we use CVXPY (Diamond and Boyd, 2016) to obtain $f^\star$, to use the (relative) suboptimality gap $\frac{f(\boldsymbol{\theta}^t)-f^\star}{f^\star}$ as a metric. We define iteration as the cumulative number of hops and denote by C_{C2C} and C_{C2S} the cost of Client-To-Client and Client-To-Server communications, respectively, whose ratio is important for SDFL. For simplicity, we assume C_{C2S} is the same for communications from the client to the server and vice-versa, although this is often not the case. Throughout Section 4, we consider $C_{C2S}/C_{C2C} = 100$ when plotting the suboptimality gap with respect to the communication cost, multiplying the number of C_{C2C} communications by 0.01 (each communication unit is the size of the token, which varies with the setup) before adding them to the number of C2S communications, to obtain the communication costs. For MTCD, we assume a uniform distribution over the clients when resuming roaming. In Section 4.1, we allow for the tokens to overlap.

We perform ridge regression on a dataset generated following the same process as (Alghunaim et al., 2021), with $N = 1000$ samples and $d = 2000$ features split evenly across clients. We have $f(\boldsymbol{\theta}) = \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2/2 + \alpha\|\boldsymbol{\theta}\|_2^2/2$, with $\alpha = 10$. For this problem, we use $\eta = 10^{-5}$ and $Q = 20$ for MTCD and $\eta = 5 \times 10^{-7}$ and $Q = 20$ for S-VFL. For DCPA, we use $\mu_w = 0.01$, $\mu_y = 0.0003$, and $\mu_x = 0.03$.

We perform sparse logistic regression on the Gisette dataset (Guyon et al., 2004), where $N = 6000$ and $d = 5000$, again split evenly across clients. Let $s(z) := (1 + e^{-z})^{-1}$ and $\beta = 1$:

$$f(\boldsymbol{\theta}) = -\sum_n \left[y_n \log s(\boldsymbol{x}_n^\top \boldsymbol{\theta}) + (1 - y_n) \log(1 - s(\boldsymbol{x}_n^\top \boldsymbol{\theta})) \right] + \beta \|\boldsymbol{\theta}\|_1, \quad y_n \in \{0, 1\},\ n \in [N].$$

where $\boldsymbol{x}_n$ and y_n denote samples and labels, respectively. We use $\eta = 10^{-4}$ and $Q = 30$ for MTCD and $\mu_w = 0.001, \mu_y = 0.00003, \mu_x = 0.003$ for DCPA.[3]

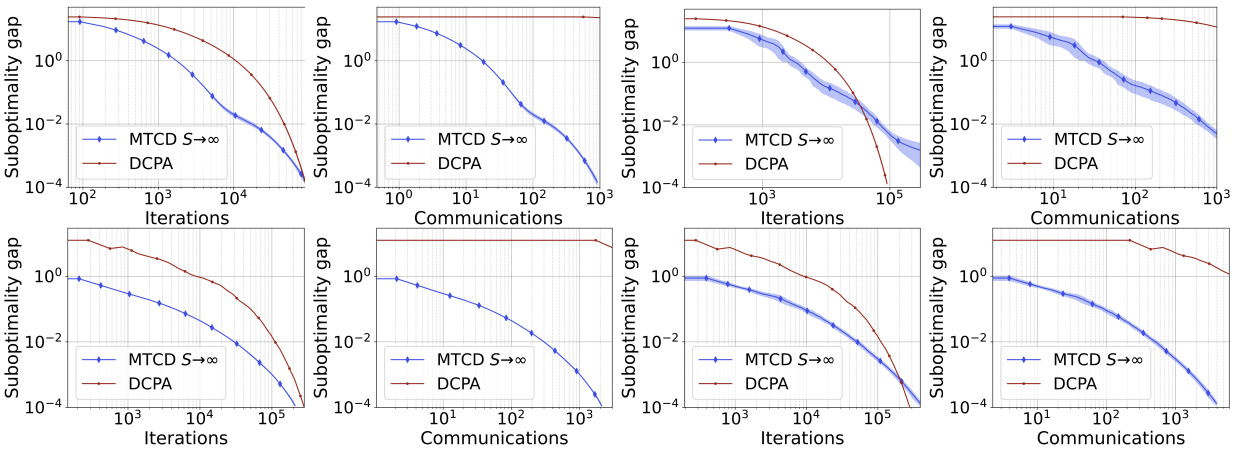


Figure 3: The left four plots correspond to an Erdős–Rényi graph with $p = 0.4$ and right four plots to a path graph, all with $K = 40$. The top row concerns ridge regression and the bottom row concerns sparse logistic regression. We consider a communication unit consisting of N scalars. The $S \to \infty$ MTCD run has $\Gamma = 1$.

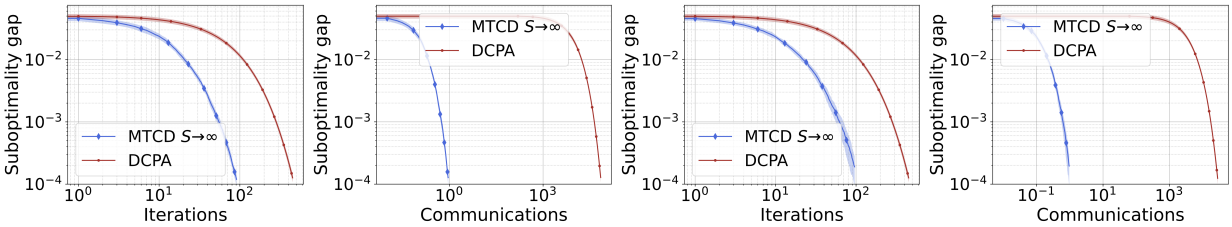
Fully decentralized setting. In Figure 3, we see that, while MTCD with $S \to \infty$ and $\Gamma = 1$ (that is, STCD) does not improve upon DCPA in terms of progress per iteration, it is significantly more communication efficient. Yet, STCD is particularly vulnerable to poorly connected networks, as seen when going from an Erdős–Rényi graph to a path graph. Note that, for sparse logistic regression, while the proximal term used to handle the regularizer is not covered by our analysis, it does well empirically. Figure 4 shows additional ridge regression experiments in the $N \gg d$ regime across six different graph topologies.

SDFL setting. We now tackle the same ridge regression problem, now focusing on path graphs, where poor connectivity is a greater challenge. In Figure 5, we see in the top row that our method improves upon the communication efficiency of the other methods. When plotting the number of communications needed to attain a suboptimality gap of 10^{-4} for each C_{C2C}/C_{C2S}, we see that, for different values of C_{C2C}/C_{C2S}, the communication efficiency of SDFL methods varies. In the bottom row, we see that, as we increase the syncing frequency, the convergence per iterations speeds up and the communication efficiency decreases, as expected. This illustrates the flexibility of our method, which allows us to choose a regime at which to operate.

4.2 Neural network training

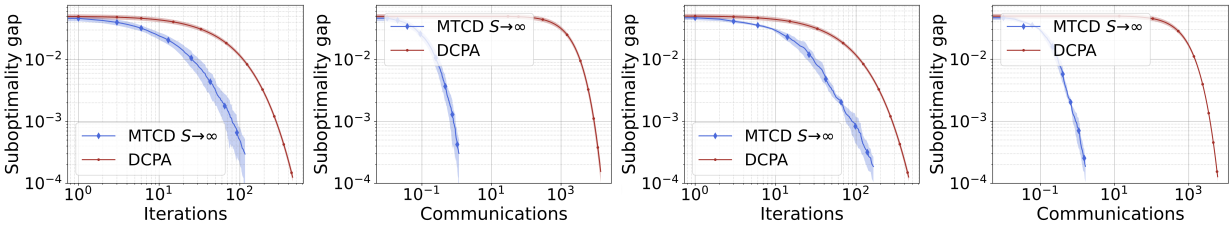
We train an MVCNN (Su et al., 2015) model on ModelNet10 (Wu et al., 2015), a dataset of 3D CAD models. We consider 12 clients split into two clusters of six clients, each capturing a different (2D) view of each object, and run MTCD for both complete and path graphs with $S = 6$. We use a fixed $\eta = 0.001$ for S-VFL. When running MTCD on 2 complete graph clusters with 6 clients each, we also start with $\eta = 0.001$ but halve it every 20 epochs. On the path graph clusters, we start with $\eta = 0.0005$ and halve it every 20 epochs. For both both types of graph, we use $S = 6$. We use $Q = 10$ and a batch size of 64 for both MTCD and S-VFL.

[3]In DCPA, the lack of a closed form solution for the proximal operator of the convex conjugate of the logistic regression loss leads to the need to solve a local optimization problem at each client at every iteration.



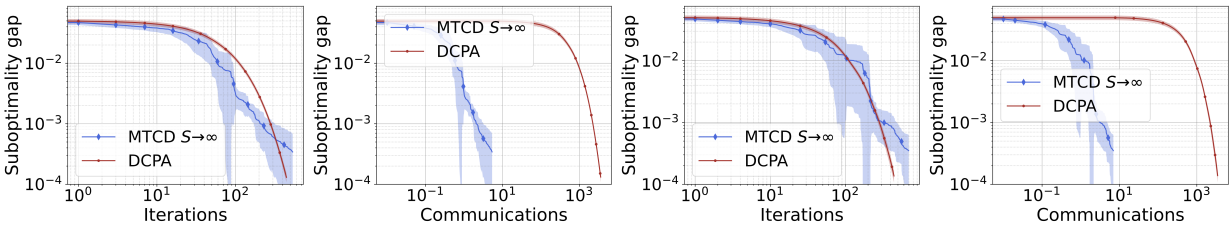
(a) Complete graph ($\alpha_{\mathcal{G}} = 40$)

(b) Erdős–Rényi graph with $p = 0.4$ ($\alpha_{\mathcal{G}} = 7.7$)



(c) Erdős–Rényi graph with $p = 0.2$ ($\alpha_{\mathcal{G}} = 1.4$)

(d) Grid graph (5×8) ($\alpha_{\mathcal{G}} = 0.15$)



(e) Cycle graph ($\alpha_{\mathcal{G}} = 0.025$)

(f) Path graph ($\alpha_{\mathcal{G}} = 0.0061$)

Figure 4: Ridge regression with $N = 4000$ and $d = 200$ across six different network topologies, all with $K = 40$ clients. Each communication unit consists of N scalars. The MTCD run has $\Gamma = 1$. Algebraic connectivity, $\alpha_{\mathcal{G}}$, is the second smallest eigenvalue of the Laplacian matrix of graph $\mathcal{G}$.

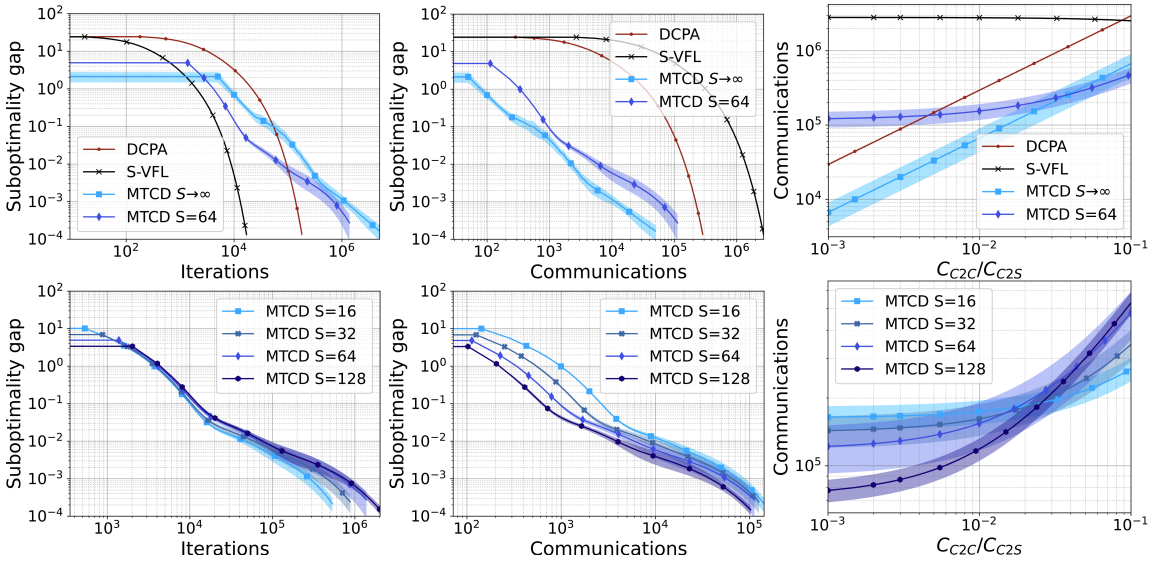


Figure 5: Experiment on a $K = 80$ path graph. For the top row, the MTCD run with $S \to \infty$ has $\Gamma = 1$ and the one with $S = 64$ has $\Gamma = 2$. For the bottom row, all MTCD runs have $\Gamma = 2$.

We also train a ResNet18 (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009), for $K = 4$ and two clusters (2 clients each). For MTCD, we use $S = 2$. We use a fixed $\eta = 0.0001$, $Q = 10$ and a batch size of 100 for both S-VFL and MTCD. For MTCD, we use $S = 2$.

In Figure 6, we present the results of the ModelNet10 and the CIFAR-10 experiments, both in the token-per-cluster setting. In both, we observe a similar performace in terms of convergence per iteration, but that MTCD outperforms the baseline in communication efficiency.

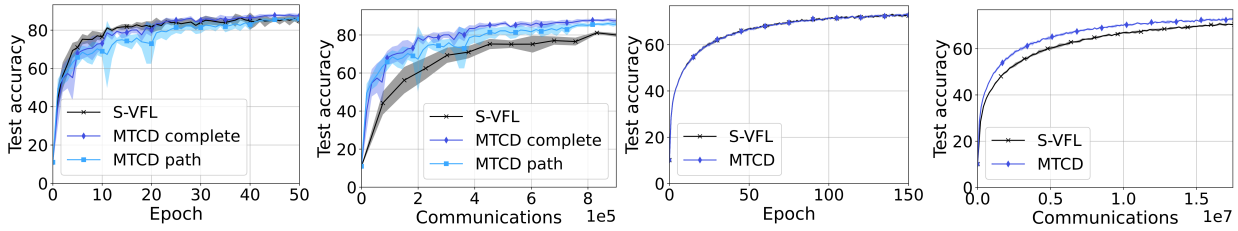


Figure 6: The two plots on the left regard ModelNet10 and the two on the right regard CIFAR-10. We use $\{h_1(\boldsymbol{\theta}_1), \ldots, h_K(\boldsymbol{\theta}_K), \boldsymbol{\theta}_0\}$ as a communication unit and, for simplicity, we assume that the embeddings and the server model have the same size.

5 Conclusions

We formalize the multi-token SDFL scheme and propose MTCD, a communication-efficient SDFL algorithm for vertical FL. We provide convergence guarantees for our method and show empirically the improved communication efficiency of our algorithm as well as the power of endowing decentralized methods with periodical client-server communications. A natural extension to this work is to consider compression and privacy mechanisms, such as differential privacy.

Acknowledgements

This work is supported in part by the Fundação para a Ciência e a Tecnologia through the Carnegie Mellon Portugal Program; by the grants U.S. National Science Foundation CCF-2007911 and ECCS-2318441; and by the CMU-Portugal project CMU/TIC/0016/2021.

References

S. A. Alghunaim, Q. Lyu, M. Yan, and A. H. Sayed. Dual consensus proximal algorithm for multi-agent sharing problems. *IEEE Transactions on Signal Processing*, 69:5568–5579, 2021.

N. Alon, C. Avin, M. Koucky, G. Kozma, Z. Lotker, and M. R. Tuttle. Many random walks are faster than one. In *Proceedings of the twentieth annual symposium on parallelism in algorithms and architectures*, pages 119–128, 2008.

A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

D. P. Bertsekas. A new class of incremental gradient methods for least squares problems. *SIAM Journal on Optimization*, 7(4):913–926, 1997.

J. Bian and J. Xu. Mobility improves the convergence of asynchronous federated learning. *arXiv:2206.04742*, 2022.

T. J. Castiglia, A. Das, S. Wang, and S. Patterson. Compressed-VFL: Communication-efficient learning with vertically partitioned data. In *International Conference on Machine Learning*, pages 2738–2766. PMLR, 2022.

I. Ceballos, V. Sharma, E. Mugica, A. Singh, A. Roman, P. Vepakomma, and R. Raskar. Splitnn-driven vertical partitioning. *arXiv preprint arXiv:2008.04137*, 2020.

H. Chen, Y. Ye, M. Xiao, and M. Skoglund. Asynchronous parallel incremental block-coordinate descent for decentralized machine learning. *arXiv:2202.03263*, 2022.

T. Chen, X. Jin, Y. Sun, and W. Yin. VAFL: a method of vertical asynchronous federated learning. *arXiv:2007.06081*, 2020.

S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

J. C. Duchi, A. Agarwal, and M. J. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606, 2012.

O. Fercoq and P. Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4), 2015.

Y. Guo, Y. Sun, R. Hu, and Y. Gong. Hybrid local SGD for federated learning with heterogeneous communications. In *International Conference on Learning Representations*, 2021.

I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge. *Advances in neural information processing systems*, 17, 2004.

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

L. He, A. Bian, and M. Jaggi. COLA: Decentralized linear learning. *Advances in Neural Information Processing Systems*, 31, 2018.

H. Hendrikx. A principled framework for the design and analysis of token algorithms. *arXiv:2205.15015*, 2022.

S. Hosseinalipour, S. S. Azam, C. G. Brinton, N. Michelusi, V. Aggarwal, D. J. Love, and H. Dai. Multi-stage hybrid federated learning over large-scale D2D-enabled fog networks. *IEEE/ACM Transactions on Networking*, 30(4):1569–1584, 2022.

B. Johansson, M. Rabi, and M. Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3):1157–1170, 2010.

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

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

D. A. Levin and Y. Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

B. Li, S. Cen, Y. Chen, and Y. Chi. Communication-efficient distributed optimization in networks with gradient tracking and variance reduction, 2020.

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

F. P.-C. Lin, S. Hosseinalipour, S. S. Azam, C. G. Brinton, and N. Michelusi. Semi-decentralized federated learning with cooperative D2D local model aggregations. *IEEE Journal on Selected Areas in Communications*, 39(12):3851–3869, 2021.

Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang. Vertical federated learning. *arXiv preprint arXiv:2211.12814*, 2022a.

Y. Liu, X. Zhang, Y. Kang, L. Li, T. Chen, M. Hong, and Q. Yang. FedBCD: A communication-efficient collaborative learning framework for distributed features. *IEEE Transactions on Signal Processing*, 70: 4277–4290, 2022b.

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*, 68:2513–2528, 2020.

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

A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

A. Nedic, A. Olshevsky, and M. G. Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

D. Paulin. Concentration inequalities for Markov chains by Marton couplings and spectral methods. 2015.

G. Qu and N. Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2018.

S. S. Ram, A. Nedić, and V. V. Veeravalli. Incremental stochastic subgradient algorithms for convex optimization. *SIAM Journal on Optimization*, 20(2):691–717, 2009.

P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2012.

V. Smith, S. Forte, C. Ma, M. Takac, M. I. Jordan, and M. Jaggi. Cocoa: A general framework for communication-efficient distributed optimization, 2018.

H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of the IEEE international conference on computer vision*, 2015.

T. Sun, Y. Sun, Y. Xu, and W. Yin. Markov chain block coordinate descent. *Computational Optimization and Applications*, 75(1):35–61, 2019.

P. Valdeira, Y. Chi, C. Soares, and J. Xavier. A multi-token coordinate descent method for vertical federated learning. In *NeurIPS Workshop on Federated Learning: Recent Advances and New Challenges*, 2022.

K. Wei, J. Li, C. Ma, M. Ding, S. Wei, F. Wu, G. Chen, and T. Ranbaduge. Vertical federated learning: Challenges, methodologies and experiments. *arXiv preprint arXiv:2202.04309*, 2022.

S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

Y. Ye, H. Chen, Z. Ma, and M. Xiao. Decentralized consensus optimization based on parallel random walk. *IEEE Communications Letters*, 24(2):391–395, 2020.

M. Yemini, R. Saha, E. Ozfatura, D. Gunduz, and A. J. Goldsmith. Semi-decentralized federated learning with collaborative relaying. In *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022.

X. Zhang, Y. Liu, J. Liu, A. Argyriou, and Y. Han. D2D-assisted federated learning in mobile edge computing networks. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021a.

X. Zhang, W. Yin, M. Hong, and T. Chen. Hybrid federated learning: Algorithms and implementation, 2021b.

H. Zhao, B. Li, Z. Li, P. Richtárik, and Y. Chi. BEER: Fast $O(1/T)$ Rate for Decentralized Nonconvex Optimization with Communication Compression. *arXiv:2201.13320*, 2022.

A Preliminaries

If a function is L-smooth (A1), then the following quadratic upper bound holds:

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2, \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d. \tag{5}$$

Definition 1 (Markov chain)**.** *A sequence of random variables $(Z^t)_{t\geq 0}$ is a finite-state space $\mathcal{K}$ Markov chain if, for all $i, j \in \mathcal{K}$,*

$$\mathbb{P}(Z^{t+1} = j \mid Z^0, \ldots, Z^{t-1}, Z^t = i) = \mathbb{P}(Z^{t+1} = j \mid Z^t = i) =: (\boldsymbol{P}(t))_{ij},$$

where $\boldsymbol{P}(t) \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{K}|}$ is called the transition matrix. A Markov chain is said to be:

- *time-homogeneous if the transition matrix is constant, that is, $\boldsymbol{P}(t) = \boldsymbol{P}$ for all t;*
- *irreducible if, starting from any state, all states can be reached eventually, that is, there exists a t such that $(\boldsymbol{P}^t)_{ij} > 0$ for all $i, j \in \mathcal{K}$;*
- *aperiodic if $gcd(\{t \geq 1 \colon (\boldsymbol{P}^t)_{ii} > 0\}) = 1$ for all states $i \in \mathcal{K}$, that is, if all states have period 1, where the period of a state i is defined as the greatest common divisor of the set of times when it is possible for the Markov chain to return to starting position i;*
- *reversible if $\pi_i \boldsymbol{P}_{ij} = \pi_j \boldsymbol{P}_{ji}$ for all $i, j \in \mathcal{K}$, where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{|\mathcal{K}|})$ is the stationary distribution, which is defined as the distribution satisfying $\boldsymbol{\pi}^\top = \boldsymbol{\pi}^\top \boldsymbol{P}$.*

A Markov chain is a lazy random walk (henceforth referred to simply as a random walk) if its states correspond to nodes of a graph and

$$\boldsymbol{P}_{ij} = \begin{cases} \frac{1}{d(i)+1} & \text{if } j \in \bar{\mathcal{N}}_i, \\ 0 & \text{otherwise,} \end{cases}$$

where $d(i)$ denote the degree of node i. A random walk on a static, connected, and undirected graph specifies a finite-state, time-homogeneous, irreducible, and aperiodic Markov chain. If a Markov chain is irreducible and aperiodic, every row of $\boldsymbol{P}^t$ will converge to a unique stationary distribution $\boldsymbol{\pi}$, as shown in (Levin and Peres, 2017). We define $\pi_{\min} \coloneqq \min_i \pi_i$ and denote by $d_{\max}$ and $d_{\min}$ the maximum and minimum degrees of any node $k \in \mathcal{K}$, respectively.

Definition 2 (Mixing time)**.** *We say τ_ϵ is the ϵ-mixing time of a Markov chain if*

$$\tau_\epsilon \coloneqq \min\left\{t \colon \sup_{i\in\mathcal{K}} d_{TV}\left(\left(\boldsymbol{P}^t\right)_{i:}, \boldsymbol{\pi}\right) \leq \epsilon\right\},$$

where d_{TV} is the total variation distance. For countable sets, our case, $d_{TV}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_1/2$.

We use the following result in our analysis.

Algorithm 3: Markov chain coordinate descent (token roaming)

Input : initial point $\boldsymbol{v}^{0,0}$, initial block k^0, step-size η, block samples S, updates per block Q

$\boldsymbol{v}^{S,Q} \leftarrow \mathcal{M}(\boldsymbol{v}^{0,0}, k^0, \eta, S, Q, \psi)$

Function $\mathcal{M}(\boldsymbol{v}, k, \eta, S, Q, \psi)$**:**

  $\boldsymbol{v}^{0,0}, k^0 \leftarrow \boldsymbol{v}, k$

  for $s = 0, \dots, S-1$ **do**

    for $q = 0, \dots, Q-1$ **do**

      $\boldsymbol{v}^{s,q+1} \leftarrow \boldsymbol{v}^{s,q} - \eta \bar{\nabla}_{k^s} \psi\left(\boldsymbol{v}^{s,q}\right)$

    $\boldsymbol{v}^{s+1,0} \leftarrow \boldsymbol{v}^{s,Q}$

    $k^{s+1} \sim \boldsymbol{P}_{k^s:}$

  return $\boldsymbol{v}^{S,Q}$

Remark (Grönwall-Bellman lemma)**.** *Let $a, b \geq 0$ be constants, if sequence $(u_n)_{n=0}^N$ satisfies*

$$u_n \leq a + b \sum_{i=0}^{n-1} u_i, \quad 0 \leq n \leq N, \qquad \text{then} \qquad u_n \leq a(1+b)^n, \quad 0 \leq n \leq N. \tag{6}$$

Proof. The $n = 0$ base case is trivial. For the induction step, we assume the result we want to show holds for $0 \leq n \leq k-1$ and notice that:

$$u_k \leq a + b \sum_{i=0}^{k-1} u_i \leq a + b \sum_{i=0}^{k-1} a(1+b)^i = a\left(1 + b \sum_{i=0}^{k-1} (1+b)^i\right) \overset{(a)}{\leq} a(1+b)^k,$$

where in (a) we use the finite sum of a geometric series, which is equal to the expression that follows for the case $b > 0$ and, to handle $b = 0$, we can use Bernoulli's inequality. □

Consider the set of vectors $\{\boldsymbol{a}_1, \dots, \boldsymbol{a}_n\}$, using the Cauchy–Schwarz inequality we have that:

$$\left\| \sum_{i=1}^{n} \boldsymbol{a}_i \right\|^2 \leq n \sum_{i=1}^{n} \|\boldsymbol{a}_i\|^2 . \tag{7}$$

B Token roaming lemmas

Lemmas 1, 2, and 3 concern token roaming. Yet, for clarity, we abstract away the tokens and communications, studying a setup-agnostic Markov chain coordinate descent instead, summarized in Algorithm 3. We consider $\psi \colon \mathbb{R}^{d'} \to \mathbb{R}$, a partitioning of $[d']$ with blocks $k \in \mathcal{K}$, and let $\bar{\nabla}_k \psi(\boldsymbol{v}) \in \mathbb{R}^{d'}$ denote a vector whose block k corresponds to $\nabla_k \psi(\boldsymbol{v})$ and all other entries are zeros. Note that k^i is fixed for $i = 0$ and a random variable for $i \in [S]$. We further define the surrogate offset as $\boldsymbol{\delta}^{i,j} \coloneqq \bar{\nabla}_{k^i} \psi\left(\boldsymbol{v}^{i,j}\right) - \nabla \psi\left(\boldsymbol{v}^{0,0}\right)$. As mentioned in the paper, token roaming is closely related to (Sun et al., 2019), yet our analysis differs significantly, since the results in (Sun et al., 2019) require $S \to \infty$ and our end goal is to study the convergence of MTCD with respect to syncing steps, for a finite number of hops S.

Lemma 1. *If $\left(\boldsymbol{v}^{i,j}\right)$ is a sequence generated by Algorithm 3, ψ is L-smooth, and $\eta \in \left(0, \frac{1}{LSQ}\right]$, then:*

$$\|\boldsymbol{\delta}_{k^s}^{s,q}\| \leq C_1 \left\|\nabla \psi\left(\boldsymbol{v}^{0,0}\right)\right\|, \qquad C_1 \coloneqq \eta e LSQ. \tag{8}$$

Proof. Let $(\triangle)$ denote the triangle inequality and $\sum_{i,j}$ denote $\sum_{i=0}^{s} \sum_{j=0}^{\tilde{q}_s(i)}$, where $\tilde{q}_s(i)$ is $q-1$ if $i = s$ and $Q-1$ otherwise, we first upper bound $\|\boldsymbol{\delta}_{k^s}^{s,q}\|$ with the sum of the norm of the blocks of the gradient used to update $\boldsymbol{v}^{0,0}$ to $\boldsymbol{v}^{s,q}$:

$$\|\boldsymbol{\delta}_{k^s}^{s,q}\| = \left\|\nabla_{k^s} \psi\left(\boldsymbol{v}^{s,q}\right) - \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right)\right\|$$

$$
\begin{aligned}
&\overset{(A1)}{\leq} L \left\| \boldsymbol{v}^{s,q} - \boldsymbol{v}^{0,0} \right\| \\
&= \eta L \left\| \sum_{i,j} \bar{\nabla}_{k^i} \psi \left(\boldsymbol{v}^{i,j} \right) \right\| \\
&\overset{(\triangle)}{\leq} \eta L \sum_{i,j} \left\| \bar{\nabla}_{k^i} \psi \left(\boldsymbol{v}^{i,j} \right) \right\| \\
&= \eta L \sum_{i,j} \left\| \nabla_{k^i} \psi \left(\boldsymbol{v}^{i,j} \right) \right\| .
\end{aligned}
$$

We now use the definition of $\boldsymbol{\delta}^{i,j}$ and the triangle inequality in order to be able to exploit the fact that $\|\nabla\psi(\boldsymbol{v}^{0,0})\|$ is fixed, allowing us to resort to the Grönwall-Bellman lemma:

$$
\begin{aligned}
\| \boldsymbol{\delta}^{s,q}_{k^s} \| &\overset{(\triangle)}{\leq} \eta L \sum_{i,j} \left(\left\| \nabla_{k^i} \psi \left(\boldsymbol{v}^{0,0} \right) \right\| + \left\| \boldsymbol{\delta}^{i,j}_{k^i} \right\| \right) \\
&\overset{(a)}{\leq} \eta L (Qs+q) \left\| \nabla \psi \left(\boldsymbol{v}^{0,0} \right) \right\| + \eta L \sum_{i,j} \left\| \boldsymbol{\delta}^{i,j}_{k^i} \right\| \\
&\leq \eta LSQ \left\| \nabla \psi \left(\boldsymbol{v}^{0,0} \right) \right\| + \eta L \sum_{i,j} \left\| \boldsymbol{\delta}^{i,j}_{k^i} \right\| \\
&\overset{(6)}{\leq} \eta LSQ \left\| \nabla \psi \left(\boldsymbol{v}^{0,0} \right) \right\| (1+\eta L)^{Qs+q} \\
&\leq \eta LSQ \left\| \nabla \psi \left(\boldsymbol{v}^{0,0} \right) \right\| (1+\eta L)^{SQ} \\
&\leq \eta eLSQ \left\| \nabla \psi \left(\boldsymbol{v}^{0,0} \right) \right\| ,
\end{aligned}
$$

where in (a) we use the fact that $\|\nabla_{k^i}\psi(\boldsymbol{v}^{0,0})\| \leq \|\nabla\psi(\boldsymbol{v}^{0,0})\|$ and in the last inequality we use our upper bound on the step-size $\eta \leq \frac{1}{LSQ}$ and the fact that $(1+\frac{1}{u})^u < e$ for all $u > 0$, thus arriving at (8). □

Lemma 2. *If $(\boldsymbol{v}^{s,q})$ is a sequence generated by Algorithm 3 and $\mathbb{P}(k^S = k) > 0$ for all k, then:*

$$
\mathbb{E} \left[\sum_{s=0}^{S} \sum_{q=0}^{Q-1} \left\| \nabla_{k^s} \psi \left(\boldsymbol{v}^{0,0} \right) \right\|^2 \right] \geq \rho Q \left\| \nabla \psi \left(\boldsymbol{v}^{0,0} \right) \right\|^2 , \tag{9}
$$

where the expectation is taken over $\{k^s\}_{s=1}^S$ and $\mathbb{P}(k^S = k) \geq \rho > 0$ for all k. We can obtain different lower bounds on ρ depending on the assumptions we make:

- *If $\boldsymbol{P}$ is a lazy random walk on a graph, then $\rho \geq (1/(d_{\max}+1))^S$;*
- *If the graph is static, connected, and undirected and $S \geq \frac{3\log(2|\mathcal{E}|/(d_{\min}+1))}{2\lambda}$, where $\lambda := 1 - \lambda_2 \in (0,1)$, with λ_2 denoting the second largest eigenvalue of $\boldsymbol{P}$ in absolute value, then $\rho \geq \frac{d_{\min}+1}{4|\mathcal{E}|}$.*

While $\mathbb{P}(k^S = k) > 0$ is an assumption needed for Lemma 2, it brings no loss of generality for the theorems where Lemma 2 is used, where this property always holds. This will become clear when we prove the theorems.

Proof. Let $\sum_{s,q}$ denote $\sum_{s=0}^{S} \sum_{q=0}^{Q-1}$, we have that

$$
\mathbb{E} \left[\sum_{s,q} \left\| \nabla_{k^s} \psi \left(\boldsymbol{v}^{0,0} \right) \right\|^2 \right] = \sum_{s,q} \sum_{k \in \mathcal{K}} \mathbb{P}(k^s = k) \left\| \nabla_k \psi \left(\boldsymbol{v}^{0,0} \right) \right\|^2 = \sum_{k \in \mathcal{K}} Q \sum_{s} \mathbb{P}(k^s = k) \left\| \nabla_k \psi \left(\boldsymbol{v}^{0,0} \right) \right\|^2 ,
$$

and, using the fact that $\left\| \nabla_k \psi \left(\boldsymbol{v}^{0,0} \right) \right\|^2 \geq 0$, we get:

$$
\sum_{k \in \mathcal{K}} Q \sum_{s} \mathbb{P}(k^s = k) \left\| \nabla_k \psi \left(\boldsymbol{v}^{0,0} \right) \right\|^2 \geq \sum_{k \in \mathcal{K}} Q \mathbb{P}(k^S = k) \left\| \nabla_k \psi \left(\boldsymbol{v}^{0,0} \right) \right\|^2 .
$$

We now obtain two different lower bounds on $\mathbb{P}(k^S = k)$, for different sets of assumptions:

- If $\boldsymbol{P}$ is a lazy random walk on a graph, $\rho \geq (1/d_{\max}+1)^S$ follows directly from the assumption that $\mathbb{P}(k^S = k) > 0$ and the transition probabilities of the random walk:

$$\mathbb{P}(k^S) = \sum_{k^1,\ldots,k^{S-2}} \left[\prod_{i=0}^{S-2} \mathbb{P}(k^{i+1} \mid k^i)\right] \geq 0 \implies \mathbb{P}(k^S = k) \geq \left(\frac{1}{d_{\max}+1}\right)^S;$$

- To avoid an exponential dependency on S we can further assume that the graph is static, connected, and undirected,[4] and that $S \geq \tau$, where τ is the $\frac{\pi_{\min}}{2}$-mixing time. It follows from the definition of mixing time (Definition 2) that $|\left(\boldsymbol{P}^S\right)_{ij} - \pi_j| \leq \pi_{\min}$ for all i, j and therefore $\left(\boldsymbol{P}^S\right)_{ij} \geq \frac{\pi_{\min}}{2}$. (If $\left(\boldsymbol{P}^S\right)_{ij} < \frac{\pi_{\min}}{2}$, then $\pi_{\min} - \left(\boldsymbol{P}^S\right)_{ij} > \frac{\pi_{\min}}{2}$, contradicting the definition of $\frac{\pi_{\min}}{2}$-mixing time.) We can now easily verify that, for a random walk, $\boldsymbol{\pi} = \left(\frac{d(k)+1}{2|\mathcal{E}|}\right)_{k\in\mathcal{V}}$, and thus:

$$\left(\boldsymbol{P}^S\right)_{k^0 k} \geq \frac{d_{\min}+1}{4|\mathcal{E}|}.$$

  We now want to upper bound τ with respect to properties of $\boldsymbol{P}$, to know for what values of S this bound holds. For a reversible, aperiodic, and irreducible Markov chain, our case, we have (Paulin, 2015, Proposition 3.3):

$$\tau \leq \frac{3\log(1/\pi_{\min})}{2\lambda} = \frac{3\log(2|\mathcal{E}|/(d_{\min}+1))}{2\lambda}.$$

We now use one of these bounds on ρ and the fact that $\sum_{k\in\mathcal{K}} \|\nabla_k \psi(\boldsymbol{v})\|^2 = \|\nabla\psi(\boldsymbol{v})\|^2$ to arrive at (9). □

Note that, in the special case of a random walk on complete graph we have $\mathbb{P}(k^s = k) = 1/|\mathcal{K}|$ for all k, allowing us to avoid the inequalities above altogether and get:

$$\mathbb{E}\left[\sum_{s=0}^{S}\sum_{q=0}^{Q-1} \left\|\nabla_{k^s}\psi\left(\boldsymbol{v}^{0,0}\right)\right\|^2\right] = \frac{SQ}{|\mathcal{K}|}\left\|\nabla\psi\left(\boldsymbol{v}^{0,0}\right)\right\|^2.$$

Lemma 3. *Assume ψ is L-smooth and let $\left(\boldsymbol{v}^{i,j}\right)$ be a sequence generated by Algorithm 3 with $\eta \in \left(0, \frac{1}{(L+1)SQ}\right]$. If*

$$\Psi := \Psi_1 + \Psi_2,$$

where

$$\Psi_1 := \left\langle \nabla\psi\left(\boldsymbol{v}^{0,0}\right), \boldsymbol{v}^{S,Q} - \boldsymbol{v}^{0,0}\right\rangle$$

and

$$\Psi_2 := \frac{L+1}{2}\left\|\boldsymbol{v}^{S,Q} - \boldsymbol{v}^{0,0}\right\|^2,$$

then

$$\Psi \leq \eta SQC_1(1+\eta(L+1)SQC_1)\left\|\nabla\psi\left(\boldsymbol{v}^{0,0}\right)\right\|^2 + \eta(\eta(L+1)SQ-1)\sum_{s=0}^{S-1}\sum_{q=0}^{Q-1}\left\|\nabla_{k^s}\psi\left(\boldsymbol{v}^{0,0}\right)\right\|^2. \tag{10}$$

Proof. Let $\sum_{s,q}$ denote $\sum_{s=0}^{S}\sum_{q=0}^{Q-1}$, we first upper bound Ψ_1 and then Ψ_2.

Bounding Ψ_1. We first use the Algorithm 3 update relating $\boldsymbol{v}^{S,Q}$ with $\boldsymbol{v}^{0,0}$ and the definition of $\boldsymbol{\delta}^{s,q}$:

$$\begin{aligned}\Psi_1 = \left\langle \nabla\psi\left(\boldsymbol{v}^{0,0}\right), \boldsymbol{v}^{S,Q} - \boldsymbol{v}^{0,0}\right\rangle &= -\eta\left\langle \nabla\psi\left(\boldsymbol{v}^{0,0}\right), \sum_{s,q}\bar{\nabla}_{k^s}\psi\left(\boldsymbol{v}^{s,q}\right)\right\rangle \\ &= -\eta\sum_{s,q}\left\langle \nabla_{k^s}\psi\left(\boldsymbol{v}^{0,0}\right), \nabla_{k^s}\psi\left(\boldsymbol{v}^{s,q}\right)\right\rangle\end{aligned}$$

[4] Similar bounds can be found for more general graphs, but we focus on this case for simplicity.

$$= -\eta \sum_{s,q} \left\langle \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right), \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right) + \boldsymbol{\delta}_{k^s}^{s,q} \right\rangle$$
$$= -\eta \sum_{s,q} \left\| \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2 - \eta \sum_{s,q} \left\langle \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right), \boldsymbol{\delta}_{k^s}^{s,q} \right\rangle.$$

Now, using first the Cauchy–Schwarz inequality and then Lemma 1, we have

$$\begin{aligned}
\Psi_1 &\leq -\eta \sum_{s,q} \left\| \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2 + \eta \sum_{s,q} \left\| \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right) \right\| \left\| \boldsymbol{\delta}_{k^s}^{s,q} \right\| \\
&\overset{(8)}{\leq} -\eta \sum_{s,q} \left\| \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2 + \eta C_1 \sum_{s,q} \left\| \nabla \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2 \\
&= -\eta \sum_{s,q} \left\| \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2 + \eta SQ C_1 \left\| \nabla \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2. \qquad (11)
\end{aligned}$$

Bounding Ψ_2. Similarly to bounding Ψ_1, we first use the update relating $\boldsymbol{v}^{S-1,Q-1}$ with $\boldsymbol{v}^{0,0}$ and the definition of $\boldsymbol{\delta}^{i,j}$

$$\begin{aligned}
\Psi_2 = \frac{L+1}{2} \left\| \boldsymbol{v}^{S,Q} - \boldsymbol{v}^{0,0} \right\|^2 &= \frac{\eta^2 (L+1)}{2} \left\| \sum_{s,q} \bar{\nabla}_{k^s} \psi\left(\boldsymbol{v}^{s,q}\right) \right\|^2 \\
&\overset{(7)}{\leq} \frac{\eta^2 (L+1) SQ}{2} \sum_{s,q} \left\| \bar{\nabla}_{k^s} \psi\left(\boldsymbol{v}^{s,q}\right) \right\|^2 \\
&= \frac{\eta^2 (L+1) SQ}{2} \sum_{s,q} \left\| \nabla_{k^s} \psi\left(\boldsymbol{v}^{s,q}\right) \right\|^2 \\
&\overset{(7)}{\leq} \eta^2 (L+1) SQ \sum_{s,q} \left(\left\| \boldsymbol{\delta}_{k^s}^{s,q} \right\|^2 + \left\| \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2 \right),
\end{aligned}$$

and then use Lemma 1, to obtain

$$\begin{aligned}
\Psi_2 &\overset{(8)}{\leq} \eta^2 (L+1) SQ \sum_{s,q} \left(C_1^2 \left\| \nabla \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2 + \left\| \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2 \right) \\
&= \eta^2 (L+1) S^2 Q^2 C_1^2 \left\| \nabla \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2 + \eta^2 (L+1) SQ \sum_{s,q} \left\| \nabla_{k^s} \psi\left(\boldsymbol{v}^{0,0}\right) \right\|^2. \qquad (12)
\end{aligned}$$

Finally, we get (10) by summing (11) and (12). □

C Proof of Theorem 1

For the sake of clarity and simplicity, we now consider a setup-agnostic stochastic disjoint parallel Markov chain coordinate descent, presented in Algorithm 4, which abstracts away the tokens and communications of MTCD while remaining mathematically equivalent. Map $\mathcal{M}$ is as defined in Algorithm 3.

In this setup, only block $\boldsymbol{\theta}_{\mathcal{C}_\gamma}(\gamma)$ of $\boldsymbol{\theta}(\gamma)$ is updated during the roaming step. Since $\mathcal{C}_1, \ldots, \mathcal{C}_\Gamma$ are disjoint, no coordinate is updated simultaneously for different model estimates. This allows us to use the simpler notation $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\mathcal{C}_1}(1), \ldots, \boldsymbol{\theta}_{\mathcal{C}_\Gamma}(\Gamma))$, focusing on the blocks of $\boldsymbol{\theta}(1), \ldots, \boldsymbol{\theta}(\Gamma)$ which are updated during the roaming step and seeing their concatenation as a single model estimate.

We define $\mathcal{V}_c^t$ as the sub-block of coordinates in block $\mathcal{C}_c$ with a nonzero probability of being updated during roaming step t, $\mathcal{V}_c^t \coloneqq \{k \in \mathcal{C}_c \colon \sum_{s=0}^{S-1} \mathbb{P}(k_c^{t,s} = k \mid k_c^{t,0}) > 0\}$. Further, we denote by f_c^t as the marginal function of f which keeps block $\mathcal{V}_c^t$ of $\boldsymbol{\theta}$ as a variable and is parameterized by the remaining coordinates of $\boldsymbol{\theta}$ after the last sync, $f_c^t(\boldsymbol{\theta}_{\mathcal{V}_c^t}^{t,s,q}) = f(\boldsymbol{\theta}_{\mathcal{V}_c^t}^{t,s,q}; \boldsymbol{\theta}_{-\mathcal{V}_c^t}^t)$. Similarly, $\mathcal{V}^t \coloneqq \bigcup_c \mathcal{V}_c^t$ and $f^t(\boldsymbol{\theta}_{\mathcal{V}^t}^{t,s,q}) = f(\boldsymbol{\theta}_{\mathcal{V}^t}^{t,s,q}; \boldsymbol{\theta}_{-\mathcal{V}^t}^t)$.

In the proof that follows, we let $\sum_s$, $\sum_q$, $\sum_c$ and $\sum_k$ denote $\sum_{s=0}^{S}$, $\sum_{q=0}^{Q}$, $\sum_{c=0}^{C}$, and $\sum_{k=0}^{K}$, respectively.

Algorithm 4: Stochastic disjoint parallel Markov chain coordinate descent

Input : initial point $\boldsymbol{\theta}^0$, step-size η, block samples S, updates per block Q

for $t = 0, \dots, T-1$ **do**
  for $c = 0, \dots, C$ **in parallel do**
    Samples batch indices $\mathcal{B}^t \subseteq [N]$
    $k_c^{t,0} \sim P_c$
    $\boldsymbol{\theta}_{\mathcal{C}_c}^{t,S,Q} \leftarrow \mathcal{M}(\boldsymbol{\theta}_{\mathcal{C}_c}^t, k_c^{t,0}, \eta, S, Q, f_c^t(\cdot; \mathcal{B}^t))$
  $\boldsymbol{\theta}^{t+1} \leftarrow (\boldsymbol{\theta}_{\mathcal{C}_0}^{t,S,Q}, \dots, \boldsymbol{\theta}_{\mathcal{C}_C}^{t,S,Q})$

Proof. It follows from L-smoothness, in particular from (5), that

$$\begin{aligned} f(\boldsymbol{\theta}^{t+1}) - f(\boldsymbol{\theta}^t) &\leq \langle \nabla f(\boldsymbol{\theta}^t), \boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t \rangle + \frac{L}{2}\|\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t\|^2 \\ &= \sum_c \left[\left\langle \nabla f_c^t \left(\boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right), \boldsymbol{\theta}_{\mathcal{V}_c^t}^{t+1} - \boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right\rangle + \frac{L}{2} \left\| \boldsymbol{\theta}_{\mathcal{V}_c^t}^{t+1} - \boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right\|^2 \right]. \end{aligned}$$

Now, let $\boldsymbol{g}_c^t \coloneqq \nabla f_c^t \left(\boldsymbol{\theta}_{\mathcal{V}_c^t}^t; \boldsymbol{X} \right) = \nabla f_c^t \left(\boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right)$ and $\tilde{\boldsymbol{g}}_c^t \coloneqq \nabla f_c^t \left(\boldsymbol{\theta}_{\mathcal{V}_c^t}^t; \mathcal{B}_t \right)$, we have

$$\begin{aligned} f(\boldsymbol{\theta}^{t+1}) - f(\boldsymbol{\theta}^t) &\leq \sum_c \left[\left\langle \tilde{\boldsymbol{g}}_c^t, \boldsymbol{\theta}_{\mathcal{V}_c^t}^{t+1} - \boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right\rangle + \left\langle \boldsymbol{g}_c^t - \tilde{\boldsymbol{g}}_c^t, \boldsymbol{\theta}_{\mathcal{V}_c^t}^{t+1} - \boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right\rangle + \frac{L}{2} \left\| \boldsymbol{\theta}_{\mathcal{V}_c^t}^{t+1} - \boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right\|^2 \right] \\ &\leq \sum_c \left[\left\langle \tilde{\boldsymbol{g}}_c^t, \boldsymbol{\theta}_{\mathcal{V}_c^t}^{t+1} - \boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right\rangle + \frac{L+1}{2} \left\| \boldsymbol{\theta}_{\mathcal{V}_c^t}^{t+1} - \boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right\|^2 + \frac{1}{2}\|\boldsymbol{g}_c^t - \tilde{\boldsymbol{g}}_c^t\|^2 \right], \end{aligned}$$

using the fact that $\langle a, b \rangle \leq \frac{1}{2}\|a\|^2 + \frac{1}{2}\|b\|^2$ on the term $\left\langle \boldsymbol{g}_c^t - \tilde{\boldsymbol{g}}_c^t, \boldsymbol{\theta}_{\mathcal{V}_c^t}^{t+1} - \boldsymbol{\theta}_{\mathcal{V}_c^t}^t \right\rangle$. We now define and assume $\zeta \coloneqq \eta(L+1)SQ < 1$ and use Lemma 3 to bound the first two terms, with ψ as $f_c^t(\cdot; \mathcal{B}_t)$:

$$f(\boldsymbol{\theta}^{t+1}) - f(\boldsymbol{\theta}^t) \leq \sum_c \left[\eta SQC_1(1 + \zeta C_1) \left\| \tilde{\boldsymbol{g}}_c^t \right\|^2 + \eta(\zeta - 1) \sum_{s,q} \left\| \tilde{\boldsymbol{g}}_{ck_c^{t,s}}^t \right\|^2 + \frac{1}{2}\|\boldsymbol{g}_c^t - \tilde{\boldsymbol{g}}_c^t\|^2 \right],$$

where $C_1 = \eta eLSQ$. Let $\sigma^{t,s}$ denote the sigma algebra $\sigma(\boldsymbol{k}^{0,0}, \dots, \boldsymbol{k}^{0,S-1}, \dots, \boldsymbol{k}^{t,0}, \dots, \boldsymbol{k}^{t,s-1})$ and $\mathbb{E}^\sigma$ be a shorthand for the conditional expectation $\mathbb{E}[\cdot \mid \sigma]$. We take the conditional expectation $\mathbb{E}^{\sigma^{t,1}}$ on both sides of the inequality above and, since $\zeta \leq 1$, it follows from Lemma 2 that

$$\begin{aligned} \mathbb{E}^{\sigma^{t,1}} \left[f(\boldsymbol{\theta}^{t+1}) \right] - f(\boldsymbol{\theta}^t) &\leq -\underbrace{\eta Q(\rho(1-\zeta) - SC_1(1+\zeta C_1))}_{=:C_2} \sum_c \left\| \tilde{\boldsymbol{g}}_c^t \right\|^2 + \frac{1}{2} \sum_c \|\boldsymbol{g}_c^t - \tilde{\boldsymbol{g}}_c^t\|^2 \\ &= -C_2 \left\| \tilde{\boldsymbol{g}}^t \right\|^2 + \frac{1}{2}\|\boldsymbol{g}^t - \tilde{\boldsymbol{g}}^t\|^2, \end{aligned}$$

where $\boldsymbol{g}^t \coloneqq \nabla f^t \left(\boldsymbol{\theta}_{\mathcal{V}^t}^t \right) = \nabla f^t \left(\boldsymbol{\theta}_{\mathcal{V}^t}^t; \boldsymbol{X} \right)$ and $\tilde{\boldsymbol{g}}^t \coloneqq \nabla f^t \left(\boldsymbol{\theta}_{\mathcal{V}^t}^t; \mathcal{B}_t \right)$. Now, taking $\mathbb{E}^{\sigma^{t,0}}$ on both sides, we have

$$\begin{aligned} \mathbb{E}^{\sigma^{t,0}} \left[f(\boldsymbol{\theta}^{t+1}) \right] - f(\boldsymbol{\theta}^t) &\leq \mathbb{E}^{\sigma^{t,0}} \left[\sum_k \mathbb{I}\{k \in \mathcal{V}^t\} \left(-C_2 \left\| \nabla_k f \left(\boldsymbol{\theta}^t; \mathcal{B}_t \right) \right\|^2 + \frac{1}{2} \left\| \nabla_k f \left(\boldsymbol{\theta}^t; \boldsymbol{X} \right) - \nabla_k f \left(\boldsymbol{\theta}^t; \mathcal{B}_t \right) \right\|^2 \right) \right] \\ &= -C_2 \sum_k \mathbb{P}(k \in \mathcal{V}^t) \left\| \nabla_k f \left(\boldsymbol{\theta}^t; \mathcal{B}_t \right) \right\|^2 + \frac{1}{2} \sum_k \mathbb{P}(k \in \mathcal{V}^t) \left\| \nabla_k f \left(\boldsymbol{\theta}^t; \boldsymbol{X} \right) - \nabla_k f \left(\boldsymbol{\theta}^t; \mathcal{B}_t \right) \right\|^2 \\ &\leq -pC_2 \left\| \nabla f \left(\boldsymbol{\theta}^t; \mathcal{B}_t \right) \right\|^2 + \frac{1}{2} \left\| \nabla f \left(\boldsymbol{\theta}^t; \boldsymbol{X} \right) - \nabla f \left(\boldsymbol{\theta}^t; \mathcal{B}_t \right) \right\|^2, \end{aligned}$$

where in the last inequality we use the fact that, for k in cluster $c(k)$, $\mathbb{P}(k \in \mathcal{V}^t) \geq \mathbb{P}(k_{c(k)}^{t,0} = k) \geq p$ to bound the first term and $\mathbb{P}(k \in \mathcal{V}^t) \leq 1$ to bound the second term. To obtain a simpler, more interpretable constant,

we bound C_2. Since $\zeta > C_1 = \eta eLSQ$, and using the fact that $\zeta \in (0,1) \implies \zeta^3 < \zeta$, we have

$$C_2 \geq \eta Q\big(\rho(1-\zeta) - Se\zeta(1+e\zeta^2)\big) = \eta Q\big(\rho - (\rho+Se)\zeta - Se^2\zeta^3\big) \geq \eta Q\big(\rho - (\rho+Se(1+e))\zeta\big) =: C_3.$$

For $\eta \in \left(0, \frac{\rho}{(L+1)SQ(\rho+Se(1+e))}\right)$, which satisfies the bounds imposed on η by the lemmas, $C_3 > 0$. We now take $\mathbb{E}^{\mathcal{B}_t} := \mathbb{E}_{\mathcal{B}_t}[\cdot \mid \mathcal{B}_0, \ldots, \mathcal{B}_{t-1}]$ and lower bound C_2 with C_3:

$$\begin{aligned}\mathbb{E}^{\mathcal{B}_t}\mathbb{E}^{\sigma^{t,0}}\left[f(\boldsymbol{\theta}^{t+1})\right] - f(\boldsymbol{\theta}^t) &\leq -pC_3\mathbb{E}^{\mathcal{B}_t}\left\|\nabla f\left(\boldsymbol{\theta}^t;\mathcal{B}_t\right)\right\|^2 + \frac{1}{2}\mathbb{E}^{\mathcal{B}_t}\left\|\nabla f\left(\boldsymbol{\theta}^t;\boldsymbol{X}\right) - \nabla f\left(\boldsymbol{\theta}^t;\mathcal{B}_t\right)\right\|^2 \\ &\leq -pC_3\left\|\nabla f\left(\boldsymbol{\theta}^t;\boldsymbol{X}\right)\right\|^2 + \left(\frac{1}{2} - pC_3\right)\mathbb{E}^{\mathcal{B}_t}\left\|\nabla f\left(\boldsymbol{\theta}^t;\boldsymbol{X}\right) - \nabla f\left(\boldsymbol{\theta}^t;\mathcal{B}_t\right)\right\|^2,\end{aligned}$$

where we used the fact that $\mathbb{E}\|X\|^2 = \|\mathbb{E}X\|^2 + \mathbb{E}\|X - \mathbb{E}X\|^2$ and (A2) on the first term. Now, noting that $\frac{1}{2} - pC_3 \geq 0$ always holds, it follows from the bounded variance assumption (A3) that

$$\mathbb{E}^{\mathcal{B}_t}\mathbb{E}^{\sigma^{t,0}}\left[f(\boldsymbol{\theta}^{t+1})\right] - f(\boldsymbol{\theta}^t) \leq -pC_3\left\|\nabla f(\boldsymbol{\theta}^t)\right\|^2 + \left(\frac{1}{2} - pC_3\right)\frac{\sigma^2}{B}.$$

We now take the expected value with respect to all $\{k_c^{t,s}\}$ and $\{\mathcal{B}_t\}$ and rearrange the terms, we get that

$$\mathbb{E}\left\|\nabla f\left(\boldsymbol{\theta}^t\right)\right\|^2 \leq \frac{1}{pC_3}\mathbb{E}\left[f\left(\boldsymbol{\theta}^t\right) - f\left(\boldsymbol{\theta}^{t+1}\right)\right] + \left(\frac{1}{2pC_3} - 1\right)\frac{\sigma^2}{B}.$$

Finally, we sum over $t \in \{0, \ldots, T-1\}$, getting a telescoping sum, and divide by T. Dropping the $-\sigma^2/B$ term, using the fact that $f^\star \leq f(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$, and defining $C_4 := \frac{1}{pC_3}$, we finally arrive at

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\left\|\nabla f\left(\boldsymbol{\theta}^t\right)\right\|^2\right] \leq \frac{C_4\Delta}{T} + \frac{C_4\sigma^2}{2B}.$$

□

D Proof of Theorem 2

For the sake of clarity and simplicity, we again consider a setup-agnostic version of our method which abstracts away the tokens and communications of MTCD while remaining mathematically equivalent. A stochastic parallel Markov chain coordinate descent where we allow for overlapping blocks of coordinates to be updated simultaneously is presented in Algorithm 5. Similarly to Appendix C, map $\mathcal{M}$ is as defined in Algorithm 3 and we define $\mathcal{V}_\gamma^t := \{k \in \mathcal{V}\colon \sum_{s=0}^{S-1}\mathbb{P}(k_\gamma^{t,s} = k \mid k_\gamma^{t,0}) > 0\}$, $f_\gamma^t(\boldsymbol{\theta}_{\mathcal{V}_\gamma^t}^{t,s,q}) = f(\boldsymbol{\theta}_{\mathcal{V}_\gamma^t}^{t,s,q}; \boldsymbol{\theta}_{-\mathcal{V}_\gamma^t}^t)$, $\mathcal{V}^t := \bigcup_\gamma \mathcal{V}_\gamma^t$, and $f^t(\boldsymbol{\theta}_{\mathcal{V}^t}^{t,s,q}) = f(\boldsymbol{\theta}_{\mathcal{V}^t}^{t,s,q}; \boldsymbol{\theta}_{-\mathcal{V}^t}^t)$.

Algorithm 5: Stochastic parallel Markov chain coordinate descent (with overlapping blocks)

Input : initial point $\boldsymbol{\theta}^0$, step-size η, block samples S, updates per block Q
for $t = 0, \ldots, T-1$ **do**
  for $\gamma = 0, \ldots, \Gamma$ **in parallel do**
    Samples batch indices $\mathcal{B}^t \subseteq [N]$
    $k_\gamma^{t,0} \sim P$
    $\boldsymbol{\theta}_{\mathcal{V}_\gamma^t}^{t,S,Q}(\gamma) \leftarrow \mathcal{M}(\boldsymbol{\theta}_{\mathcal{V}_\gamma^t}^t, k_\gamma^{t,0}, \eta, S, Q, f_\gamma^t(\cdot;\mathcal{B}^t))$
  $\boldsymbol{\theta}^{t+1}(1), \ldots, \boldsymbol{\theta}^{t+1}(\Gamma) = \frac{1}{\Gamma}\sum_{\gamma=1}^{\Gamma}\boldsymbol{\theta}^{t,S,Q}(\gamma)$

The following proof resembles that of Theorem 1, yet we now use the convexity assumption to handle the averaging of model estimates. We let $\sum_s$, $\sum_q$, and $\sum_\gamma$ denote $\sum_{s=0}^S$, $\sum_{q=0}^Q$, and $\sum_{\gamma=0}^\Gamma$, respectively.

Proof. It follows from the convexity assumption (A4) that:

$$f(\boldsymbol{\theta}^{t+1}) - f(\boldsymbol{\theta}^t) = f\left(\frac{1}{\Gamma}\sum_{\gamma}\boldsymbol{\theta}^{t,S,Q}(\gamma)\right) - f\left(\boldsymbol{\theta}^t\right) \leq \frac{1}{\Gamma}\sum_{\gamma}\left(f\left(\boldsymbol{\theta}^{t,S,Q}(\gamma)\right) - f\left(\boldsymbol{\theta}^t\right)\right).$$

Therefore, resorting to L-smoothness, in particular (5), we have that,

$$\begin{aligned} f(\boldsymbol{\theta}^{t+1}) - f(\boldsymbol{\theta}^t) &\leq \frac{1}{\Gamma}\sum_{\gamma}\left(\left\langle \nabla f(\boldsymbol{\theta}^t), \boldsymbol{\theta}^{t,S,Q}(\gamma) - \boldsymbol{\theta}^t\right\rangle + \frac{L}{2}\|\boldsymbol{\theta}^{t,S,Q}(\gamma) - \boldsymbol{\theta}^t\|^2\right) \\ &= \frac{1}{\Gamma}\sum_{\gamma}\left(\left\langle \nabla f_\gamma^t(\boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}), \boldsymbol{\theta}^{t,S,Q}_{\mathcal{V}^t_\gamma}(\gamma) - \boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}\right\rangle + \frac{L}{2}\|\boldsymbol{\theta}^{t,S,Q}_{\mathcal{V}^t_\gamma}(\gamma) - \boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}\|^2\right) \end{aligned}$$

and, defining $\boldsymbol{g}^t_\gamma \coloneqq \nabla f^t_\gamma\left(\boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}; \boldsymbol{X}\right) = \nabla f^t_\gamma\left(\boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}\right)$ and $\tilde{\boldsymbol{g}}^t_\gamma \coloneqq \nabla f^t_\gamma\left(\boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}; \mathcal{B}_t\right)$, we arrive at

$$\begin{aligned} f(\boldsymbol{\theta}^{t+1}) - f(\boldsymbol{\theta}^t) &\leq \frac{1}{\Gamma}\sum_{\gamma}\left(\left\langle \tilde{\boldsymbol{g}}^t_\gamma, \boldsymbol{\theta}^{t,S,Q}_{\mathcal{V}^t_\gamma}(\gamma) - \boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}\right\rangle + \left\langle \boldsymbol{g}^t_\gamma - \tilde{\boldsymbol{g}}^t_\gamma, \boldsymbol{\theta}^{t,S,Q}_{\mathcal{V}^t_\gamma}(\gamma) - \boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}\right\rangle + \frac{L}{2}\|\boldsymbol{\theta}^{t,S,Q}_{\mathcal{V}^t_\gamma}(\gamma) - \boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}\|^2\right) \\ &\leq \frac{1}{\Gamma}\sum_{\gamma}\left(\left\langle \tilde{\boldsymbol{g}}^t_\gamma, \boldsymbol{\theta}^{t,S,Q}_{\mathcal{V}^t_\gamma}(\gamma) - \boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}\right\rangle + \frac{L+1}{2}\|\boldsymbol{\theta}^{t,S,Q}_{\mathcal{V}^t_\gamma}(\gamma) - \boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}\|^2 + \frac{1}{2}\|\boldsymbol{g}^t_\gamma - \tilde{\boldsymbol{g}}^t_\gamma\|^2\right), \end{aligned}$$

where we used $\langle a, b\rangle \leq \frac{1}{2}\|a\|^2 + \frac{1}{2}\|b\|^2$ on the term $\left\langle \boldsymbol{g}^t_\gamma - \tilde{\boldsymbol{g}}^t_\gamma, \boldsymbol{\theta}^{t,S,Q}_{\mathcal{V}^t_\gamma}(\gamma) - \boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma}\right\rangle$. We now define and assume $\zeta \coloneqq \eta(L+1)SQ < 1$ and use Lemma 3 to bound the first two terms, with ψ as $f^t_\gamma(\cdot; \mathcal{B}_t)$:

$$\begin{aligned} f(\boldsymbol{\theta}^{t+1}) - f(\boldsymbol{\theta}^t) &\leq \frac{1}{\Gamma}\sum_{\gamma}\left(\eta SQC_1(1+\zeta C_1)\left\|\tilde{\boldsymbol{g}}^t_\gamma\right\|^2 + \eta(\zeta-1)\sum_{s,q}\left\|\tilde{\boldsymbol{g}}^t_{\gamma k^{t,s}_\gamma}\right\|^2 + \frac{1}{2}\|\boldsymbol{g}^t_\gamma - \tilde{\boldsymbol{g}}^t_\gamma\|^2\right) \\ &\leq \eta SQC_1(1+\zeta C_1)\left\|\tilde{\boldsymbol{g}}^t\right\|^2 + \frac{\eta(\zeta-1)}{\Gamma}\sum_{\gamma,s,q}\left\|\tilde{\boldsymbol{g}}^t_{\gamma k^{t,s}_\gamma}\right\|^2 + \frac{1}{2}\|\boldsymbol{g}^t - \tilde{\boldsymbol{g}}^t\|^2, \end{aligned}$$

where $\boldsymbol{g}^t \coloneqq \nabla f^t\left(\boldsymbol{\theta}^t_{\mathcal{V}^t}\right) = \nabla f^t\left(\boldsymbol{\theta}^t_{\mathcal{V}^t}; \boldsymbol{X}\right)$ and $\tilde{\boldsymbol{g}}^t \coloneqq \nabla f^t\left(\boldsymbol{\theta}^t_{\mathcal{V}^t}; \mathcal{B}_t\right)$ and the last inequality follows from the fact that $\|\nabla f^t(\boldsymbol{\theta}^t_{\mathcal{V}^t})\|^2 \leq \sum_\gamma \|\nabla f^t_\gamma(\boldsymbol{\theta}^t_{\mathcal{V}^t_\gamma})\|^2$. We take the conditional expectation $\mathbb{E}^{\sigma^{t,1}}$ on both sides of the inequality above and, since $\zeta \leq 1$, it follows from Lemma 2 that:

$$\begin{aligned} \mathbb{E}^{\sigma^{t,1}}\left[f(\boldsymbol{\theta}^{t+1})\right] - f(\boldsymbol{\theta}^t) &\leq \eta SQC_1(1+\zeta C_1)\left\|\nabla f^t(\boldsymbol{\theta}^t_{\mathcal{V}^t})\right\|^2 + \eta\rho' Q(\zeta-1)\left\|\nabla f^t(\boldsymbol{\theta}^t_{\mathcal{V}^t})\right\|^2 + \frac{1}{2}\|\boldsymbol{g}^t - \tilde{\boldsymbol{g}}^t\|^2 \\ &= -\underbrace{\eta Q\left(\rho'(1-\zeta) - SC_1(1+\zeta C_1)\right)}_{C'_2}\left\|\nabla f^t(\boldsymbol{\theta}^t_{\mathcal{V}^t})\right\|^2 + \frac{1}{2}\|\boldsymbol{g}^t - \tilde{\boldsymbol{g}}^t\|^2. \end{aligned}$$

Note that, while in Theorem 1 the graph considered in Lemma 2 was that of each cluster, we now consider the whole communication graph. The rest of the proof matches the proof of Theorem 1, the only difference being the fact that p is replaced by p', which is such that $\mathbb{P}(k \in \mathcal{V}^t) \geq \mathbb{P}(k^{t,0}_\gamma = k) \geq p'$. Thus, for $\eta \in \left(0, \frac{\rho'}{(L+1)SQ(\rho' + Se(1+e))}\right)$, we get the inequality that we set out to prove:

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\left\|\nabla f\left(\boldsymbol{\theta}^t\right)\right\|^2\right] \leq \frac{C'_4\Delta}{T} + \frac{C'_4\sigma^2}{2B},$$

where $C'_4 \coloneqq \frac{1}{p'C'_3} > 0$, with $C'_3 \coloneqq \eta Q\big(\rho' - (\rho' + Se(1+e))\zeta\big)$. □