9-13-2022

Decentralized personalized federated learning: Lower bounds and optimal algorithm for all personalization modes

Abdurakhmon Sadiev
Moscow Institute of Physics and Technology (MIPT), Russian Federation & Mohamed bin Zayed University of Artificial Intelligence

Ekaterina Borodich
Moscow Institute of Physics and Technology (MIPT), Russian Federation & HSE University, Russian Federation

Aleksandr Beznosikov
Moscow Institute of Physics and Technology (MIPT), Russian Federation & Mohamed bin Zayed University of Artificial Intelligence & HSE University, Russian Federation

Darina Dvinskikh
HSE University, Russian Federation

Saveliy Chezhegov
Moscow Institute of Physics and Technology (MIPT), Russian Federation

Recommended Citation

A. Sadiev, et al, "Decentralized personalized federated learning: Lower bounds and optimal algorithm for all personalization modes", *EURO Journal on Computational Optimization,* vol. 10, September 2022, doi:10.1016/j.ejco.2022.100041

EURO Journal on Computational Optimization 10 (2022) 100041

Contents lists available at ScienceDirect

EURO Journal on Computational Optimization

journal homepage: www.elsevier.com/locate/ejco

Decentralized personalized federated learning: Lower bounds and optimal algorithm for all personalization modes

Abdurakhmon Sadiev [a,b,*], Ekaterina Borodich [a,c], Aleksandr Beznosikov [a,b,c], Darina Dvinskikh [c], Saveliy Chezhegov [a], Rachael Tappenden [d], Martin Takáč [b], Alexander Gasnikov [a,b,c,e]

[a] *Moscow Institute of Physics and Technology (MIPT), Russia*
[b] *Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), United Arab Emirates*
[c] *HSE University, Russia*
[d] *University of Canterbury, New Zealand*
[e] *Institute for Information Transmission Problems RAS, Russia*

ARTICLE INFO

Dataset link: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Keywords:
Federated learning
Decentralized optimization
Distributed optimization
Lower and upper bounds
Accelerated algorithms

ABSTRACT

This paper considers the problem of decentralized, personalized federated learning. For centralized personalized federated learning, a penalty that measures the deviation from the local model and its average, is often added to the objective function. However, in a decentralized setting this penalty is expensive in terms of communication costs, so here, a different penalty — one that is built to respect the structure of the underlying computational network — is used instead. We present lower bounds on the communication and local computation costs for this problem formulation and we also present provably optimal methods for decentralized personalized federated learning.

* Corresponding author.
E-mail address: sadiev.aa@phystech.edu (A. Sadiev).

https://doi.org/10.1016/j.ejco.2022.100041

Numerical experiments are presented to demonstrate the practical performance of our methods.

1. Introduction

Today's data revolution is transforming the world, with vast amounts of data collected daily from a wide range of sources. Automation is necessary when processing and extracting information from such large quantities of data, and machine learning has proven to be a useful tool to assist with this task. The essence of machine learning is to build, and then train, models. To speed up the process of training, modern computer architectures can be used, where instead of one single computing device, the problem and associated data is shared among many devices/agents. This leads to the following distributed learning/optimization problem formulation

$$\min_{x\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(x), \tag{1}$$

where each agent/device $i \in [n] := \{1, 2, \ldots, n\}$, has an associated local loss function $f_i : \mathbb{R}^d \to \mathbb{R}$, as well as its own locally stored data.

Federated Learning (FL) [15,18] is a subset of distributed machine learning, where one assumes that computing agents are simply general user devices, (for example, smartphones, tablets, laptops, personal computers), and where different devices may have different memory capacity and computing power. This leads to many new and important problems and questions that did not arise previously in the classical distributed setting [16]. For example, data may be spread unequally between devices, privacy considerations may prohibit the sharing of data between certain devices on the network, poor or unreliable connectivity may inhibit the flow of data, and data on certain devices may be of poorer quality compared with others.

Given these issues, this work focuses on *personalization* for federated learning. Notice that in (1), the model parameter x is found using global data (from all agents). However, the inclusion of a particular individual agent might negatively impact the global training process if their local data differs markedly from the global data, or if they have low quality local data; in this case training the global model may lead to a poor solution. On the other hand, each user may have very little local data, and the process of training a model solely on local data may also lead to a poor quality solution. The question of how to balance the two extremes, *global versus local*, gave rise to *Personalized Federated Learning (PFL)* [10,8,11].

For PFL, each agent i has their own parameter x_i, but the discrepancy between the parameters held on different devices is penalized. Correspondingly, PFL can be formulated as the following regularized optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^{nd}} \sum_{i=1}^{n} f_i(x_i) + \frac{\lambda}{2} r(\mathbf{x}), \tag{2}$$

where the vector $\mathbf{x} = [x_1^T, \ldots, x_n^T]^T \in \mathbb{R}^{nd}$ is the concatenation of the n local vectors $\{x_i \in \mathbb{R}^d\}_{i=1}^n$, $r(\mathbf{x})$ is a convex penalty function, and the weight parameter λ balances the degree of personalization. There are many possible choices for the penalty function $r(\mathbf{x})$. A simple option is to let $r(\mathbf{x})$ be the deviation between the local models and their average [8,10,11]:

$$r(\mathbf{x}) = \sum_{i=1}^{n} \|x_i - \bar{x}\|^2, \quad \text{where } \bar{x} = \frac{1}{n}(x_1 + \cdots + x_n). \tag{3}$$

This is a reasonable choice in the centralized distributed setting, where devices communicate with a central server, sending and receiving information without failures. In this case, calculating the average $\bar{x}$ is easy: all agents i simply send their local x_i to the central server, which then calculates the average $\bar{x}$, and communicates it back to every agent. In this paper we consider a more general setup, where different penalty functions might be more appropriate.

Throughout this work we consider *decentralized distributed learning*, where there is no main server (node), but instead all devices are connected via some large network. Moreover, each agent in the network can only communicate with its neighbors. Mathematically, the network is represented by a fixed, un-directed, connected graph, where each node corresponds to an agent, and connections between agents are represented by edges. Although a decentralized setting is assumed here, our problem formulation is general enough to include a centralized set-up as a special case (simply take a complete graph). However, a decentralized setup perhaps better captures the federated learning setting, where each device only communicates with a limited number of other agents, corresponding to an incomplete graph. As previously mentioned, communication links between certain agents may be inaccessible, for example, due to the (poor) quality of connection between agents, or due to remoteness of location, and this leads to missing edges in the graph.

In a decentralized setting, using the penalty in (3) is not sensible because of the impracticality of calculating the average $\bar{x}$. (Note that to calculate $\bar{x}$, all local x_i's must be sent to one device (node) and then the average $\bar{x}$ broadcast back to every node, which is a long and expensive operation, especially for large networks.) With this in mind, here we propose the use of a different penalty $r(\mathbf{x})$, which is more suitable for a decentralized setup. Hence, the problem formulation considered in this work is:

$$\min_{\mathbf{x}\in\mathbb{R}^{nd}} F(\mathbf{x}) = \underbrace{\sum_{i=1}^{n} f_i(x_i)}_{f(\mathbf{x})} + \underbrace{\frac{\lambda}{2}\langle \mathbf{x}, W\mathbf{x}\rangle}_{g(\mathbf{x})}, \tag{4}$$

where W is a communication matrix that reflects the properties of the network (see Section 1.2 for a formal definition of W). The function $r(\mathbf{x}) = \langle \mathbf{x}, W\mathbf{x}\rangle$ penalizes the difference between neighboring local models in the network, and is computationally friendlier than (3) in a decentralized setting. The matrix W determines how much an agent depends on each of the other nodes in the learning process. This is achieved due to the fact that W represents the structure of the communication graph, gives information about the remoteness of the nodes, the speed of transfer between them, and carries weights of how much to rely on one or another neighbor in the network. Note that $W\mathbf{x} = 0$ (and consequently $r(\mathbf{x}) = 0$) if and only if $x_1 = \cdots = x_n$. This penalty function is not new and has been used in the literature in several contexts, for example, for classical decentralized minimization with large λ [17,7,3], and for multitask PFL with small λ [23,26,3].

The parameter λ balances the 'global vs local' trade-off. For example, consider the following extremes:

* If $\lambda = 0$, then (4) becomes $\min_{\mathbf{x}\in\mathbb{R}^{nd}} \sum_{i=1}^{n} f_i(x_i)$, where the local function f_i held by agent i is minimized by x_i^*, and x_i^* is likely to be different than that obtained for agent j. This is equivalent to independent local training of the models.
* As $\lambda \to +\infty$, (4) tends to the distributed problem where the local arguments are constrained to be equal: i.e., $\min_{x_1=\cdots=x_n\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(x_i)$. This is equivalent to problem (1) and the training of one global model.

1.1. Preliminaries

Throughout this work the following assumption is made regarding the functions in (4).

Assumption 1. It is assumed that each $f_i : \mathbb{R}^d \to \mathbb{R}$ in problem (4) is

* L-smooth w.r.t the ℓ_2-norm, i.e. for all $u, v \in \mathbb{R}^d$, $\|\nabla f_i(u) - \nabla f_i(v)\|_2 \leq L\|u - v\|_2$; and
* μ-strongly-convex w.r.t. the ℓ_2-norm, i.e. $\forall u, v \in \mathbb{R}^d$, $f_i(u) - f_i(v) \geq \langle \nabla f_i(v), u - v\rangle + \frac{\mu}{2}\|u - v\|_2^2$.

By Assumption 1, f in (4) is L-smooth and μ-strongly convex, and subsequently F is also μ-strongly convex.

1.2. Communication

The communication network is modeled as a fixed, connected, undirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ are vertices (devices) and $\mathcal{E} = \{(i, j) \,|\, i, j \in \mathcal{V}\}$ are edges (connections between devices). Note that $(i, j) \in \mathcal{E}$ if and only if there exists a communication link between agents i and j. For such a graph, a gossip matrix $\hat{W}$ is defined as follows.

Definition 1 *(Gossip matrix).* A matrix $\hat{W} \in \mathbb{R}^{d\times d}$, associated with a graph $\mathcal{G}$, is called a gossip matrix, if it satisfies the following conditions:

1. $\hat{W}$ is symmetric positive semi-definite;
2. The kernel of $\hat{W}$ consists of the vector $\mathbf{1} = (1, \ldots, 1)^\top$;
3. $\hat{W}$ is defined on the edges of the communication network: $\hat{w}_{i,j} \neq 0$ if and only if $i = j$ or $(i, j) \in \mathcal{E}$.

The communication matrix W in (4) is $W = \hat{W} \otimes I_d$, i.e., W is the Kronecker product of a gossip matrix $\hat{W}$ and the identity matrix I_d. Because only neighboring agents can communicate in this decentralized optimization setting, it is assumed that communication is made via a gossip protocol [4,19], i.e., $\hat{W}$ is a gossip matrix (Definition 1) and communication is realized via matrix-vector multiplication with W. During one communication/communication round, for every node, full local vectors of dimension d (e.g. variables $\{x_i\}$ or gradients $\{\nabla f_i(x_i)\}$) are exchanged with all neighbors. This work supposes that the network remains unchanged, all connections are stable, and no interruptions nor asynchronous/delayed transmissions are considered.

Here, $\lambda_{\max}(W)$ denotes the maximum eigenvalue of W, $\lambda^+_{\min}(W)$ denotes the minimum positive eigenvalue of W and $\chi \geq \lambda_{\max}(W)/\lambda^+_{\min}(W)$ is an upper bound on the condition number. Because $W = \hat{W} \otimes I_d$, it holds that $\lambda_{\max}(W) = \lambda_{\max}(\hat{W})$ and $\lambda^+_{\min}(W) = \lambda^+_{\min}(\hat{W})$. The quantity χ reflects how quickly information is transmitted through the graph; a small χ corresponds to fast transmission, while a large χ corresponds to slow transmission.

A simple example of a matrix $\hat{W}$ satisfying Definition 1 is the Laplacian matrix. For example, the Laplacian of a linear graph (chain) is

$$\hat{W} = \begin{pmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 1 \end{pmatrix}.$$

In terms of personalization, this means that the model on the first node relies directly on the 2nd node. In turn the 2nd node depends on the 1st and 3rd nodes, and so on. In particular, the 1st and last nodes depend on each other weakly and only indirectly through the whole chain.

However, it is also possible to define $\hat{W}$ in a more complex way. For example, in the case of a linear graph, one can add weights that represent how much a given node relies upon its neighbors:

$$\hat{W} = \begin{pmatrix} 1 & -1 & & & & & & & \\ -1 & 1.5 & -0.5 & & & & & & \\ & -0.5 & 1 & -0.5 & & & & & \\ & & -0.5 & 1.2 & -0.7 & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & -0.5 & 1.5 & -1 & & \\ & & & & & -1 & 1 & -1 & \\ & & & & & & -1 & 1 & \end{pmatrix}.$$

In this example, the second node trusts the 1st node more than the 3rd, while the third node trusts the 2nd and 4th nodes equally, etc.

2. Contributions

In this paper, we study the personalized federated learning formulation (4). Lower complexity bounds for communication and local computation are proposed, and we develop several algorithms capable of achieving the lower bounds. Our results extend the work in [8], which used the penalty (3), to problem (4), which involves a penalty more amenable to the decentralized setting. Our contributions are summarized now.

– *Lower bounds.* We present lower bounds for the decentralized personalized federated learning problem (4) in the deterministic case (i.e., when we have access to full gradients for each f_i); see Section 3. The lower bounds are valid for all values of the parameter λ. In particular, in the smooth strongly convex case with small λ, the lower bounds are of the order $\sqrt{\lambda\lambda_{\max}(W)/\mu}$, which can be a significant improvement on the bound $\sqrt{\chi L/\mu}$ in the general, non-personalized case, [21]. This reflects a key advantage of the formulation (4), because it is then possible to both solve the problem of personalizing the models, and also to significantly reduce the total number of communications. This is an important factor not only in federated learning, but also in general distributed learning. Note that the lower bounds obtained in the work [8] are a special case of our lower bounds, when the communication network is represented by a fully connected graph. A summary of these lower bounds is presented in Table 1.

Table 1
Summary of complexity results (upper and lower bounds) on communications (`comm`) and local computations (`local`) for finding an ε-optimal solution of (4) in the deterministic (gradient) case.

	Lower bounds	Upper bounds
`comm`	$\widetilde{\Omega}\left(\min\left\{\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}, \sqrt{\frac{L}{\mu}\chi}\right\}\right)$	$\widetilde{\mathcal{O}}\left(\min\left\{\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}, \sqrt{\frac{L}{\mu}\chi}\right\}\right)$
`local`	$\widetilde{\Omega}\left(\sqrt{\frac{L}{\mu}}\right)$	$\widetilde{\mathcal{O}}\left(\sqrt{\frac{L}{\mu}}\right)$

– *Near-optimal algorithm.* Another contribution is the development of optimal algorithms that match the theoretical lower bounds. The Accelerated Meta-Algorithm of [6] (for general composite problems), is used as the base algorithm. The application of this algorithm to our problem formulation (4) is discussed, and specific implementation modes are suggested depending on small and large values of the regularization parameter λ. The analysis of the convergence in these modes shows that using this approach we achieve the lower optimal bounds up to logarithmic factors (Section 4.2). Hence, our algorithm is 'near-optimal' in the deterministic case; see Section 4.
– *Stochastic case.* We extend the previously reported results from the deterministic case (when the full gradient for all f_i is available), to the stochastic setting. In particular, we consider the case when each local function f_i is a finite sum (for example, the sum of batches), i.e. $f_i = \frac{1}{M}\sum_{m=1}^{M} f_{i,m}$. In this case, for one call of the oracle we can get only the gradient of one term $f_{i,m}$. We provide lower bounds, as well as a stochastic modification of our near-optimal deterministic algorithm; see Section 4.3.
– *Experiments.* We present numerical experiments to demonstrate the benefits of our approach. In particular, we used several datasets from the benchmark LIBSVM library, and we considered several different graph structures. We also run the experiments for several values of the penalty parameter λ, to better understand the impact of personalization; see Section 5.

3. Lower bounds

In this section, optimal algorithms for problems of the form (4) are described, and lower bounds on the local computation and communication costs for such optimal algorithms, are presented. We begin with the following assumption, which describes the properties of algorithms relevant for this work, (i.e., the properties of the algorithms for which the lower bounds, developed later in this section, are valid). Such an assumption is common in the literature; see, for example, [8,12,21].

Assumption 2. Consider an Algorithm $\mathcal{A}$, for problem (4). Then, the iterates $\{\mathbf{x}^k\}_{k=1}^K$ of Algorithm $\mathcal{A}$ are generated using only components available in local memory, where, for each node of graph $\mathcal{G}$ the sequence of local memory $\{\mathcal{M}_{i,k}\}_{k=1}^K$ for $1 \leq i \leq n$ is:

$$\mathcal{M}_{i,0} = \left\{x_i^0\right\},$$

$$\mathcal{M}_{i,k+1} = \begin{cases} \text{span}\left\{\mathcal{M}_{i,k}, \nabla f_i(y_i)\right\}, \forall y_i \in \mathcal{M}_{i,k} & \text{if local comp. at iteration } k \\ \text{span}\left\{\bigcup\limits_{j:(i,j)\in\mathcal{E}} \mathcal{M}_{j,k}\right\} & \text{if communication at iteration } k. \end{cases}$$

Assumption 2 can be interpreted as follows. Initially, each agent i (corresponding to a node on graph $\mathcal{G}$) has local memory $\mathcal{M}_{i,0}$, which comprises of the initial point x_i^0. At any iteration $k \geq 1$, the algorithm can either perform a computation using the locally available memory, or it can carry out a communication step. If the algorithm performs a local computation, then each device can calculate the gradient at any point from its current memory $\mathcal{M}_{i,k}$ and take a linear combination of this gradient with the previously generated points stored in $\mathcal{M}_{i,k}$. If the algorithm performs a communication step, then information is exchanged with neighbors and the current local memory $\mathcal{M}_{i,k}$ is combined with that held by its neighbors. Such an algorithm is first order, because it generates its iterates using linear combinations of local points and gradients.

We are now ready to present our first theorem, which gives a lower bound on the number of communications needed by an algorithm whose iterates are generated according to Assumption 2. (The proof can be found in Appendix B.)

Theorem 1. *Let $\chi \geq 3$, $L \geq 2\mu$, and $\lambda\lambda_{\min}^{+}(W) \geq \mu$. Then there exist functions $f_1, f_2, \ldots, f_n : \mathbb{R}^d \to \mathbb{R}$ satisfying Assumption 1, a graph $\mathcal{G}$ with associated matrix $\hat{W}$ satisfying Definition 1, and an initial point $\mathbf{x}^0 = [(x_1^0)^T, \ldots (x_n^0)^T]^T \in \mathbb{R}^{nd}$, such that any algorithm $\mathcal{A}$ (satisfying Assumption 2) among K iterations need to make at least*

$$\Omega\left(\min\left\{\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}, \sqrt{\frac{(L-\mu)\chi}{\mu}}\right\}\log\frac{1}{\varepsilon}\right) \quad \textit{communications}$$

to achieve ε-optimal solution in the outputs ($\|x_j - x_j^*\|^2 \varepsilon$ for all j).

The proof of this Theorem is placed in Appendix B.

It remains to develop lower bounds for the local computation costs for any algorithm $\mathcal{A}$ satisfying Assumption 2. Hence, consider a special instance of problem (4), where $\mathbf{x}^0 \in \mathbb{R}^{nd}$, $f_1 = f_2 = \cdots = f_n$, and $\hat{W}$ is the Laplace matrix for a fully connected graph. Then (4) reduces to the minimization of the single local function f_1 (communication is unnecessary, irrespective of λ, because the functions are all identical). Now, if f_1 is chosen to be the worst-case quadratic from [20], then the lower bound of at least

$$N^{\text{loc}} = \Omega\left(\sqrt{\tfrac{L}{\mu}}\log\tfrac{1}{\varepsilon}\right) \tag{5}$$

gradient calls are needed to find an ε-optimal solution.

4. Algorithms

The goal of this section is to develop an optimal algorithm for problem (4), i.e., to develop an algorithm whose iterates satisfy the lower bounds in Section 3. In Section 4.1, we discuss an algorithm that can be applied to general composite optimization problems. In Section 4.2, this algorithm is specialized to the application considered in this work, that of decentralized personalized federated learning (4). The algorithms in Sections 4.1 and 4.2 can be applied to deterministic problems, and the extension to a stochastic setting is considered in Section 4.3. In particular, the case when the function at each node has finite sum structure is considered, and two approaches, both equipped with convergence results, are described and compared.

4.1. Accelerated meta-algorithm

In this section, consider the general composite optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^{nd}} H(\mathbf{x}) = h_1(\mathbf{x}) + h_2(\mathbf{x}). \tag{6}$$

(Later we will consider how h_1 and h_2 in (6) are related to f and g in (4).) The following assumption is made about problem (6).

Assumption 3. For the problem (6), it is assumed that h_1 is convex and $L(h_1)$-smooth, that h_2 is convex and $L(h_2)$-smooth, and that H is μ-strongly convex.

There are many efficient algorithms that can be applied to problem (6), including the Accelerated-Meta-Algorithm (see Algorithm 1) proposed in [6], as well as its restarted version (see Algorithm 2).

Algorithm 1 Accelerated Meta-Algorithm (MA) [6].

Input: starting point $\mathbf{x}^0 \in \mathbb{R}^{nd}$, no. of iterations K, parameter $\gamma > 0$, accuracy $\delta > 0$
Initialization: $A^0 = 0, \quad \mathbf{y}^0 = \mathbf{x}^0, \quad \tau = \frac{1}{2\gamma}$
for $k = 0, \ldots, K-1$ **do**
$a^{k+1} = \dfrac{\tau + \sqrt{\tau^2 + 4\tau A^k}}{2}$
$A^{k+1} = A^k + a^{k+1}$
$\mathbf{w}^k = \dfrac{A^k}{A^{k+1}}\mathbf{y}^k + \dfrac{a^{k+1}}{A^{k+1}}\mathbf{x}^k$
Find $\mathbf{y}^{k+1} \in \mathbb{R}^{nd}$, such that $\|\hat{\mathbf{y}}^{k+1} - \mathbf{y}^{k+1}\|_2^2 \leq \delta$, where

$$\hat{\mathbf{y}}^{k+1} = \operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^{nd}} \left\{ \langle \nabla h_1(\mathbf{w}^k), \mathbf{y} - \mathbf{w}^k \rangle + h_2(\mathbf{y}) + \frac{\gamma}{2}\|\mathbf{y} - \mathbf{w}^k\|_2^2 \right\} \tag{7}$$

$\mathbf{x}^{k+1} = \mathbf{x}^k - a^{k+1}\nabla H(\mathbf{y}^{k+1})$
end for
Output: $\mathbf{y}^K$

Algorithm 2 Restarted Accelerated Meta-Algorithm (`Restarted-MA`) [6].

Input: initial point $\mathbf{x}^0 \in \mathbb{R}^{nd}$, no. of iterations S, parameter $\gamma > 0$, accuracy $\delta > 0$
Initialization: $N^s = \max\left\{\left\lceil 4 \cdot \sqrt{\frac{2\gamma}{\mu}} \right\rceil, 1\right\}$
for $s = 0, \ldots, S-1$ **do**
  $\mathbf{x}^{s+1} = \texttt{MA}(\mathbf{x}^s, N^s, \gamma, \delta)$
end for
Output: $\mathbf{x}^S$

Remark 1. Note that (7) can be solved, for example, by Accelerated Gradient Descent [20].

The following convergence results hold for `Restarted-MA` (Algorithm 2) applied to problem (6).

Theorem 2 *(Theorem 3 in [24]). Let Assumption 3 hold, let $\gamma \geq 2L(h_1)$, let $\varepsilon > 0$, and let*

$$\delta \leq \frac{\varepsilon\mu}{864^2(L(h_1) + L(h_2) + \gamma)^2}.$$

If Algorithm 2 runs for

$$S = \mathcal{O}\left(\sqrt{\frac{L(h_1)}{\mu}} \log \frac{1}{\varepsilon}\right) \tag{8}$$

iterations, generating output $\mathbf{x}^S$, then $H(\mathbf{x}^S) - H(\mathbf{x}^*) \leq \varepsilon$, where $\mathbf{x}^*$ denotes the optimal solution to (6).

4.2. Convergence analysis – near-optimal algorithm

Section 4.1 introduced an accelerated algorithm for the general problem (6), with associated convergence results. The purpose of this section is to make a connection between the results in Section 4.1, and how they are applicable in the context of personalized federated learning (i.e., problem (4)). Moreover, the lower bounds established in Section 3 related to local computation and communication costs for an optimal algorithm for problem (4). Thus, another goal is to show that Algorithms 1+2 is an optimal algorithm for (4), by showing that it achieves the lower bounds on communication and local computation costs presented in Section 3.

By comparing problems (4) and (6), it can be seen that they are both convex and composite. The key here is that we do not make a one-to-one correspondence between (f, g) and (h_1, h_2). That is, depending on the parameter λ, two different cases — one in which $f \equiv h_1$ and $g \equiv h_2$, while the other in which $f \equiv h_2$ and $g \equiv h_1$ — are considered. Practical versions of Algorithm 1 for the problem (4) are presented in Appendix A (Algorithms 4 and 5).

Regardless, to apply Algorithm 1+2, it is necessary to compute the gradients for both h_1 and h_2 (recall subproblem (7)), and therefore for both f and g when extending to the original problem (4). So, let us study how to compute the gradients ∇f and ∇g for (4), and try to understand where the communications arise. Note that the computation of ∇f does not require communication. Indeed, each block i has a corresponding gradient ∇f_i (taken with respect to the local variables x_i), and the 'long' gradient ∇f is simply the concatenation of the block gradients. On the other hand, $\nabla g(\mathbf{x}) = \lambda W\mathbf{x}$, and to compute the matrix-vector product $W\mathbf{x}$ requires communication with neighbors (recall the gossip protocol described in Section 1.2, and see also [4,19]). It can be shown that computing $\nabla g(\mathbf{x})$ is equivalent to one communication. Therefore, if we know how many times $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ are called by Algorithm 1, then complexities for the number of local computations and communications, respectively, can be obtained.

We are now ready to present the main convergence theorem of this paper, which provides complexity results for the local computation and communication costs for an optimal algorithm for problem (4).

Theorem 3. *Let Assumption 1 hold and let the graph $\mathcal{G}$ have an associated matrix $\hat{W}$ that satisfies Definition 1. Then, to obtain an ε-optimal solution to problem (4), solving by Algorithm 2 with*

$$\delta = \frac{\varepsilon\mu}{3000^2(L + \lambda\lambda_{\max}(W))^2}, \tag{9}$$

requires the number of communications and local computations, respectively, to be of the order

$$N^{comm} = \mathcal{O}\left(\min\left\{\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}, \sqrt{\frac{L}{\mu}\chi}\right\}\log\frac{1}{\varepsilon}\log\frac{1}{\delta}\right), \tag{10}$$

and

$$N^{loc} = \mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}\log\frac{1}{\delta}\right). \tag{11}$$

Proof. First, note that $\mathcal{G}$ is a quadratic function with a positive semi-definite Hessian, so it is $\lambda_{\max}(W)$-smooth and convex. Moreover, it is $\lambda\lambda^{+}_{\min}(W)$-strongly convex on the subspace $(\mathbf{Ker}\ W)^{\perp}$. By Assumption 1, f is L-smooth and μ-strongly convex. Hence, F is strongly convex. Thus, the conditions of Theorem 2 hold, and the application of its analysis is valid. The remainder of the analysis is split into two cases.

Case 1: $\lambda\lambda_{\max}(W) \geq L$. Here, let $h_1(\mathbf{x}) = f(\mathbf{x})$ and $h_2(\mathbf{x}) = g(\mathbf{x})$. Theorem 2 gives the complexity for the function $h_1 = f$, i.e. the number of local computations N^{loc} is given in (11). Also, it can be shown that δ in (9) satisfies the condition in Theorem 2.

Next, consider the auxiliary problem (7). By Definition 1, **Ker** W is not empty, and the function $g(\mathbf{x})$ takes a zero on this subspace. Then we can divide our problem into two subproblems: minimization of a quadratic form with matrix $\gamma \cdot I$ on **Ker** W and

minimization of a quadratic form with matrix $\lambda W + \gamma \cdot I$ on $(\mathbf{Ker}\ W)^{\perp}$. The complexity of the first problem is $\mathcal{O}(1)$. The second problem is $\lambda\lambda^{+}_{\min}(W)$-strongly convex, and if the Accelerated Gradient Method [20] is used to solve this subproblem, then the complexity is

$$\mathcal{O}\left(\sqrt{\frac{\gamma+\lambda\lambda_{\max}(W)}{\max\{\gamma,\lambda\lambda^{+}_{\min}(W)\}}}\log\frac{1}{\delta}\right). \tag{12}$$

This is the complexity for a single subproblem (7) solve, but (7) is solved (5) times. Overall, this means that the total number of calls of ∇g is:

$$N^{\text{comm}} = \mathcal{O}\left(\sqrt{\frac{L}{\mu}}\sqrt{\frac{\gamma+\lambda\lambda_{\max}(W)}{\max\{\gamma,\lambda\lambda^{+}_{\min}(W)\}}}\log\frac{1}{\varepsilon}\log\frac{1}{\delta}\right).$$

Noting that

$$\sqrt{\frac{\gamma+\lambda\lambda_{\max}(W)}{\max\{\gamma,\lambda\lambda^{+}_{\min}(W)\}}} = \min\left\{\sqrt{\frac{\gamma+\lambda\lambda_{\max}(W)}{\gamma}}, \sqrt{\frac{\gamma+\lambda\lambda_{\max}(W)}{\lambda\lambda^{+}_{\min}(W)}}\right\},$$

and taking $\gamma = 2L$, gives (10).

Case 2: $\lambda\lambda_{\max}(W) < L$. Here, let $h_1 = g$ and $h_2 = f$. Theorem 2 gives the complexity for the function $h_1 = g$, i.e. the number of communications is

$$N^{\text{comm}} = \mathcal{O}\left(\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}\log\frac{1}{\varepsilon}\right) = \mathcal{O}\left(\min\left\{\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}, \sqrt{\frac{L}{\mu}\chi}\right\}\log\frac{1}{\varepsilon}\log\frac{1}{\delta}\right). \tag{13}$$

In last step we additionally use that $\chi \geq 1$. Also, it can be shown that δ in (9) satisfies the condition in Theorem 2. If the Accelerated Gradient Method [20] is used to solve subproblem (7), the complexity for a single subproblem solve is again given by (12), and this subproblem is solved (13) times. Then we can find the number of calls for ∇f:

$$N^{\text{loc}} = \mathcal{O}\left(\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}\sqrt{\frac{L+\gamma}{\mu+\gamma}}\log\frac{1}{\varepsilon}\log\frac{1}{\delta}\right).$$

Taking $\gamma = 2\lambda\lambda_{\max}(W)$ gives (11).

Finally, combining the two cases establishes the theorem statement. □

Remark 2. Note that in the centralized case (with a completely connected communication network) we have that $\chi = 1,\ \lambda_{\max}(W) = 1$ and our method converges with the following rates:

$$N^{comm} = \widetilde{\mathcal{O}}\left(\min\left\{\sqrt{\frac{\lambda}{\mu}}, \sqrt{\frac{L}{\mu}}\right\}\right), \quad N^{loc} = \widetilde{\mathcal{O}}\left(\sqrt{\frac{L}{\mu}}\right).$$

These bounds coincide with lower bounds for centralized PFL [8].

4.3. Stochastic case

Here we extend the work previously presented and consider the stochastic case of problem (4). In particular, it is assumed that each local function has a sum structure, so that (4) becomes

$$\min_{\mathbf{x}\in\mathbb{R}^{nd}} \sum_{i=1}^{n} \underbrace{\frac{1}{M}\sum_{m=1}^{M} f_{i,m}(x_i)}_{f_i(x_i)} + \frac{\lambda}{2}\langle \mathbf{x}, W\mathbf{x}\rangle. \tag{14}$$

This setup often arises when we consider $f_i(x_i) = \mathbb{E}_{\xi_i\sim\mathcal{D}_i}[f_{\xi_i}(x_i)]$, where $\mathcal{D}_i$ is an unknown distribution, $f_{\xi_i}(x_i)$ represents the loss of model x_i on sample ξ_i, and $f_i(x_i)$ is the generalization error. Since we do not know the distribution $\mathcal{D}_i$, we cannot work with $f_i(x_i)$ directly, and typically replace it with an approximation via Monte Carlo integration $f_i(x_i) = \frac{1}{M}\sum_{m=1}^{M} f_{i,m}(x_i)$. In this context, the problem is known as empirical risk minimization. This formulation is currently the main setting for solving supervised learning problems [22]. Usually it is expensive to compute the full gradients $\nabla f_i(x_i)$ at each iteration, so instead, each node independently and uniformly chooses an index (batch number) m_i and calculates the gradient $\nabla f_{i,m_i}(x_i)$ for that batch only. It turns out that we obtain the stochastic gradient typical of learning processes. Moreover $\nabla f_{i,m_i}(x_i)$ is an unbiased estimator of $\nabla f_i(x_i)$.

The following assumption (a modification of Assumption 1) is used here.

Assumption 4. It is assumed that each $f_i : \mathbb{R}^d \to \mathbb{R}$ in problem (14) is:

* L-average smooth w.r.t. l_2-norm, i.e. $\forall u, v \in \mathbb{R}^d$, $\frac{1}{M}\sum_{m=1}^{M} \|\nabla f_{i,m}(u) - \nabla f_{i,m}(v)\|^2 \leq L^2\|u - v\|^2$;
* μ-strongly-convex w.r.t. l_2-norm i.e. $\forall u, v \in \mathbb{R}^d$, $\langle \nabla f_i(u) - \nabla f_i(v), u - v\rangle \geq 2\mu\|u - v\|^2$.

We present two approaches for solving problem (14). These approaches are efficient in the case of small λ. The key idea of the first approach (which uses the Accelerated Meta-Algorithm combined with L-Katyusha as the subproblem solver) is that problem (14) is considered as composite problem (6). In the second approach (Accelerated Randomized Algorithm for Decentralized Minimization) the ideas of variance reduction and importance sampling are used.

Accelerated meta-algorithm + L-Katyusha

As previously mentioned, the main idea behind this approach is to view problem (14) as the composite problem (6). In particular, Section 4.2 showed that (4) can be solved by the Accelerated Meta-Algorithm with $h_1 = g$, $h_2 = f$. With this choice of h_1 and h_2 communications occur only in the outer loop, when we compute $\nabla g(\mathbf{x}) = \lambda W\mathbf{x}$. The local computations of $\nabla f_i(x_i)$ take place in the inner loop. But now, the inner problem (7) has

a finite-sum structure (since f_i has a finite-sum structure and hence h_2 does as well). As previously mentioned, it is computationally expensive to use the full gradient for h_2, so typically for the subproblem (7), stochastic methods, such as the classical SGD method, are employed. Note that SGD converges only to a neighborhood of the solution, but for the finite-sum type problem it is known that one can use a variance reduction technique [13,1,9] to achieve convergence to an exact solution. For this reason, we chose to use an accelerated and practical method that incorporates a variance reduction approach – L-Katyusha [9].

Theorem 4. *Let Assumption 4 hold and let the graph $\mathcal{G}$ have an associated matrix $\hat{W}$ that satisfies Definition 1. Then, to obtain an ε-optimal solution to problem (4), solving by Algorithm 1 combined with L-Katyusha, with*

$$\delta = \frac{\varepsilon\mu}{3000^2(L + \lambda\lambda_{\max}(W))^2},$$

requires the number of communications and local computations, respectively, to be of the order

$$N^{comm} = \mathcal{O}\left(\sqrt{\tfrac{\lambda\lambda_{\max}(W)}{\mu}}\log\tfrac{1}{\varepsilon}\log\tfrac{1}{\delta}\right),$$

and

$$N^{loc} = \mathcal{O}\left(\left(M\sqrt{\tfrac{\lambda\lambda_{\max}(W)}{\mu}} + \sqrt{\tfrac{ML}{\mu}}\right)\log\tfrac{1}{\varepsilon}\log\tfrac{1}{\delta}\right).$$

The proof of this theorem is similar to the proof of Theorem 3 and can be found in Appendix C.

Remark 3. Note that the Accelerated Meta-Algorithm + L-Katyusha is suboptimal when $M\lambda\lambda_{\max}(W) \leq L$.

Accelerated randomized algorithm for decentralized minimization

In contrast with the previous approach, Algorithm 3 uses variance reduction and importance sampling techniques and is based on L-Katyusha [9]. We now view problem (14) as being the sum of $M + 1$ functions: there are M functions f_i, as well as the composite term g. In Line 3 of Algorithm 3 the value of a random variable ξ^k determines what to choose: f (make a local computation with probability $1 - p$) or g (make a communication with probability p). If the outcome is a local computation, then we choose index i of the function f. We give a practical version of Algorithm 3 in Appendix A (Algorithm 6).

At each iteration of the algorithm, between 0 and 2 communications are made. As noted above, the first communication can take place if $\xi^k = 0$. And then the value of a

Algorithm 3 Accelerated Randomized for Decentralized Minimization (ARDM).

Input: starting point $\mathbf{x}_0 \in \mathbb{R}^{nd}$, number of iterations K, parameters $0 < \theta_1, \theta_2 < 1$, $\eta, \beta, \gamma > 0$, probabilities p, ρ

Initialization: $\mathbf{y}_0 = \mathbf{z}_0 = \mathbf{u}_0 = \mathbf{x}_0$ and $\hat{\mathbf{g}}^0 = \lambda W \mathbf{y}^0 + \nabla f(\mathbf{y}^0)$

for $k = 0, 1, 2, \dots K-1$ **do**
  $\mathbf{x}^k = \theta_1 \mathbf{z}^k + \theta_2 \mathbf{u}^k + (1 - \theta_1 - \theta_2)\mathbf{y}^k$
  Generate $\xi^k = \begin{cases} 1, & \text{with probability } 1-p \\ 0, & \text{with probability } p \end{cases}$
  if $\xi^k = 0$ **then**
    $\mathbf{g}^k = \frac{\lambda}{p}\left(W\mathbf{x}^k - W\mathbf{u}^k\right) + \hat{\mathbf{g}}^k$
  else
    Sample indices $m_1^k, \dots m_n^k$ for each node independently and uniformly from $[M]$
    $\mathbf{g}^k = \frac{1}{1-p}\left(\nabla f_{m^k}(\mathbf{x}^k) - \nabla f_{m^k}(\mathbf{u}^k)\right) + \hat{\mathbf{g}}^k$ with $\nabla f_{m^k}(\mathbf{x}) = (\nabla f^T_{1,m_1^k}(x_1), \dots, \nabla f^T_{n,m_n^k}(x_n))^T$
  end if
  $\mathbf{y}^{k+1} = \mathbf{x}^k - \eta \mathbf{g}^k$
  $\mathbf{z}^{k+1} = \beta \mathbf{z}^k + (1-\beta)\mathbf{x}^k + \frac{\gamma}{\eta}(\mathbf{y}^{k+1} - \mathbf{x}^k)$
  Generate $\xi^{k+\frac{1}{2}} = \begin{cases} 1, & \text{with prob. } 1-\rho \\ 0, & \text{with prob. } \rho \end{cases}$
  if $\xi^{k+1/2} = 0$ **then**
    $\mathbf{u}^{k+1} = \mathbf{y}^{k+1}$
    $\hat{\mathbf{g}}^{k+1} = \lambda W \mathbf{y}^{k+1} + \nabla f(\mathbf{y}^{k+1})$
  else
    $\mathbf{u}^{k+1} = \mathbf{u}^k$
    $\hat{\mathbf{g}}^{k+1} = \hat{\mathbf{g}}^k$
  end if
end for

random variable $\xi^{k+\frac{1}{2}}$ determines whether to update $\hat{\mathbf{g}}^k$ or not. If $\hat{\mathbf{g}}^k$ is updated, then Algorithm 3 makes a communication and a local computation. The following theorem states the convergence rate of Algorithm 3.

Theorem 5. *Let Assumption 4 hold and let the graph $\mathcal{G}$ have an associated matrix $\hat{W}$ that satisfies Definition 1. Then, to obtain an ε-optimal solution to problem (4) using Algorithm 3, we can choose parameters γ, η, β, $p = \frac{\lambda\lambda_{\max}(W)}{L+\lambda\lambda_{\max}(W)}$, and $\rho = \frac{1}{M}$ such that we need the following number of communications (on average)*

$$N^{comm} = \mathcal{O}\left(\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}\log\frac{1}{\varepsilon}\right).$$

For $\rho = p$ we can achieve the following number of local computations (on average)

$$N^{loc} = \mathcal{O}\left(\left(M + \sqrt{\frac{ML}{\mu}}\right)\log\frac{1}{\varepsilon}\right).$$

The proof of this Theorem can be found in Appendix D.

Remark 4. Accelerated Meta-Algorithm + L-Katyusha has optimal local computational complexity when $M\lambda\lambda_{\max}(W) \leq L$. In contrast, the second algorithm has better local computation complexity (on average) if $\lambda\lambda_{\max}(W) < L$ and $M\lambda\lambda_{\max}(W) \geq L$.

Table 2
The number of features and number of samples for each dataset used in the numerical experiments.

dataset	# features (d)	# samples
mushrooms	112	8,124
a9a	123	32,561
covtype.scale	54	581,012
rcv1.binary	47,236	20,242

5. Numerical experiments

In this section, we present several numerical experiments to demonstrate the practical advantages of the proposed approach for problem (4). We study logistic loss functions,

$$f_i(x_i) = \frac{1}{n}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\log(1+e^{-y_j^i a_j^i x_i})\right),$$

where $\{(a_j^i, y_j^i)\}_{j=1}^{m_i}$ is local dataset stored on each machine $i \in \{1, 2, \ldots, n\}$, $a_j^i \in \mathbb{R}^d$ represents the feature vector and $y_j^i \in \{-1, 1\}$ is the label. In the experiments, the power method was used to estimate the smoothness parameter of the objective function, as well as $\lambda_{\max}(W)$.

Datasets The experiments were performed on datasets from the LIBSVM [5] database.[1] Table 2 shows the basic characteristics of the datasets that were used.

The communication networks In the experiments, three different network topologies were considered:

1. *Cyclic:* In this topology, devices are connected in a cycle, where each device is connected to it's two closest neighbors only. In this communication network, it takes $\sim \frac{n}{2}$ iterations to transmit information between two devices on opposite sides of the cycle.
2. *Grid:* Here devices are organized in a $\sqrt{n} \times \sqrt{n}$ grid, and are connected to their nearest neighbors.[2]
3. *Erdos:* A random communication graph, also known as an Erdős-Rényi graph.[3]

[1] The datasets are available at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.
[2] https://networkx.org/documentation/networkx-1.10/reference/generated/networkx.generators.classic.grid_2d_graph.html.
[3] https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.erdos_renyi_graph.html.

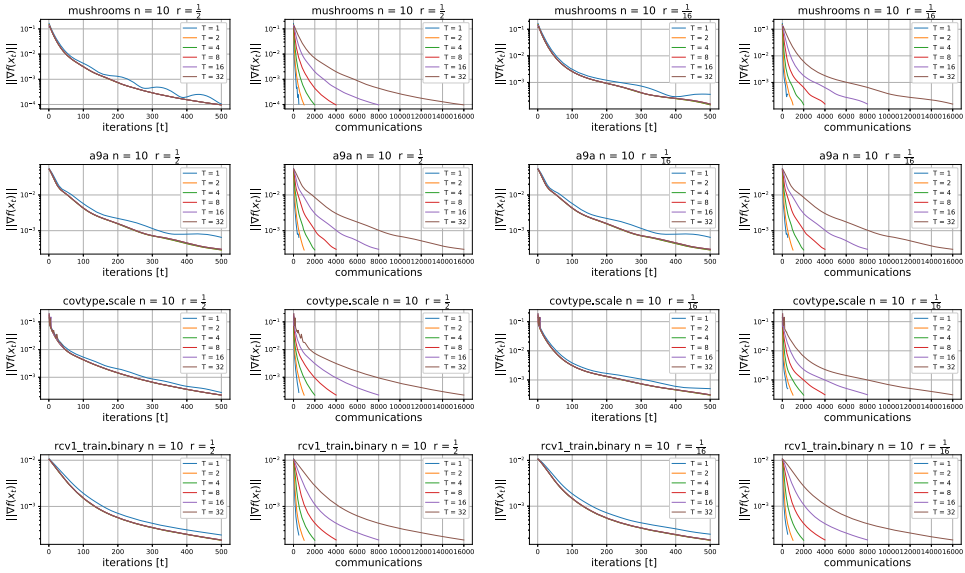

Fig. 1. Evolution of $\|\nabla f(x_t)\|$ for different datasets, regularization parameter and different level of solving the subproblem (7) (larger T means we optimize the subproblem better).

We used the **networkx** python package[4] to generate random bi-directional graphs with the structures described above. As highlighted in the theory, the algorithm depends on the parameters λ, $\lambda_{\max}(W)$ and L. We ran several experiments with varying values of λ, where

$$\lambda = r\frac{L}{\lambda_{\max}(W)}, \qquad r > 0. \tag{15}$$

5.1. Solving the sub-problem

Algorithm 1 requires the solution to the auxiliary problem (7). To avoid communication costs, an approximate solution to (7) was obtained by performing T iterations of Nesterov's accelerated gradient method. In Fig. 1 we show the evolution of $\|\nabla f(x_t)\|$ for various selections of parameter T. Observe that the behavior for $T \in \{2, 4, \ldots, 32\}$ is almost identical (in terms of the iterations of the algorithm), however, larger T requires additional rounds of communications. Therefore, in the following experiments we selected $T = 2$.

5.2. Effect of the regularization parameter

The main benefit of personalized federated learning is the ability to have slightly different local models, x_i, for each device i. The regularization term $\lambda x^T W x$ penalizes

[4] The **networkx** package https://networkx.org/ is hosted at https://github.com/networkx/networkx.

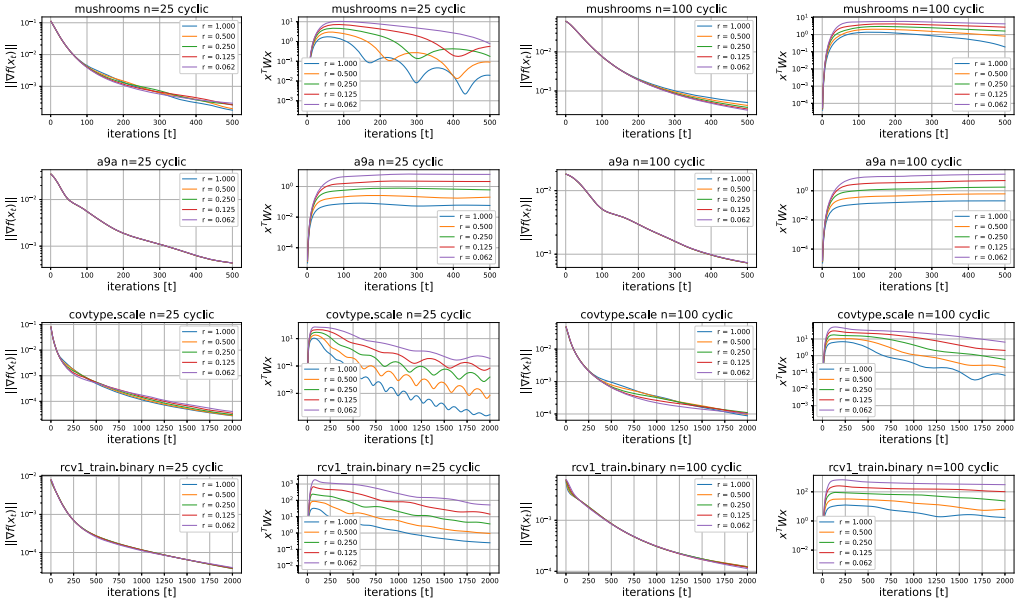

Fig. 2. Evolution of $\|\nabla f(x_t)\|$ and $x_t^T W x_t$ for various datasets, levels of λ and number of devices $n \in \{25, 100\}$ with cyclic network.

local models (i.e., the x_i's) that are different from their mean, where the parameter λ controls the emphasis placed on this penalty term. When λ is large, problem (4) tends to a consensus/classical federated learning problem, because there is a large penalty for models that are different at distinct devices. The current work focuses on *personalized* federated learning, so here we consider the small λ regime.

Recall that for each problem instance considered, two iterations of accelerated gradient descent ($T = 2$) were used to give an approximate solution to subproblem (7). The parameter λ is defined in (15), and several values of $r \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$ were used. Let us stress that, as the number of local functions n increases, the matrix W changes, and hence, so too does $\lambda_{\max}(W)$. One can observe that, as expected, larger values of λ (that corresponds to larger values of r) lead to solutions $x \in \mathbb{R}^{nd}$ that have a smaller value of the penalty term $x^T W x$. Figs. 2 (cyclic network), 3 (grid network) and 4 (Erdős-Rényi network) show the results of several numerical experiments.

5.3. Local training accuracy

In Fig. 5 we demonstrate the main benefit of using PFL - namely, the ability for each device to have a slightly different local model, thereby capturing small differences in the local data. This is done by selecting various values of λ and observing the affect that has on the training accuracy over various local functions f_i. We plot the average accuracy over local accuracies (each using their own set of parameters). For the *mushrooms* and *a9a* datasets, the algorithm quickly achieved very good local accuracy for all

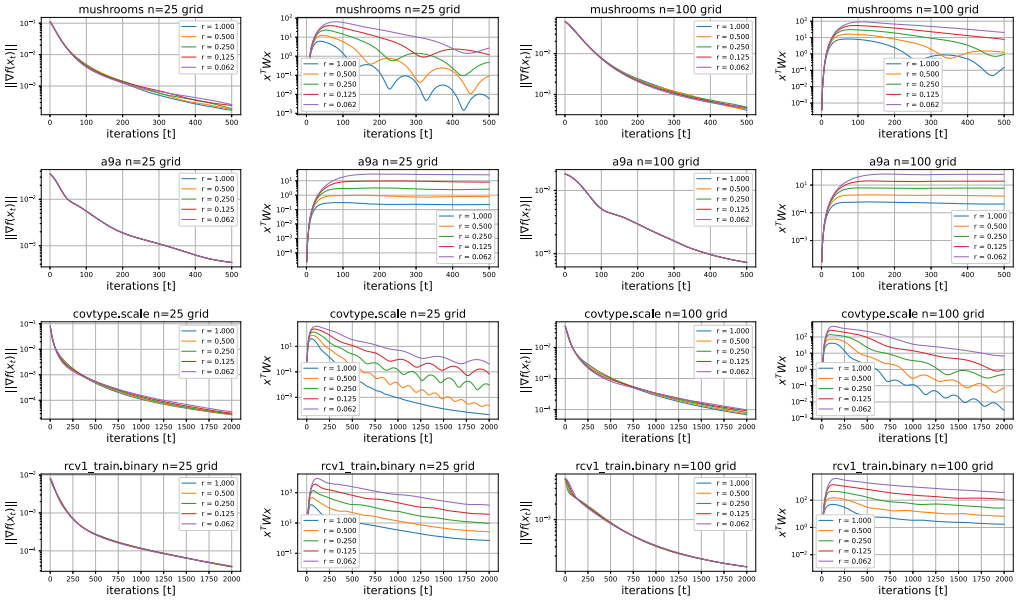

Fig. 3. Evolution of $\|\nabla f(x_t)\|$ and $x_t^T W x_t$ for various datasets, levels of λ and number of devices $n \in \{25, 100\}$ with grid network.

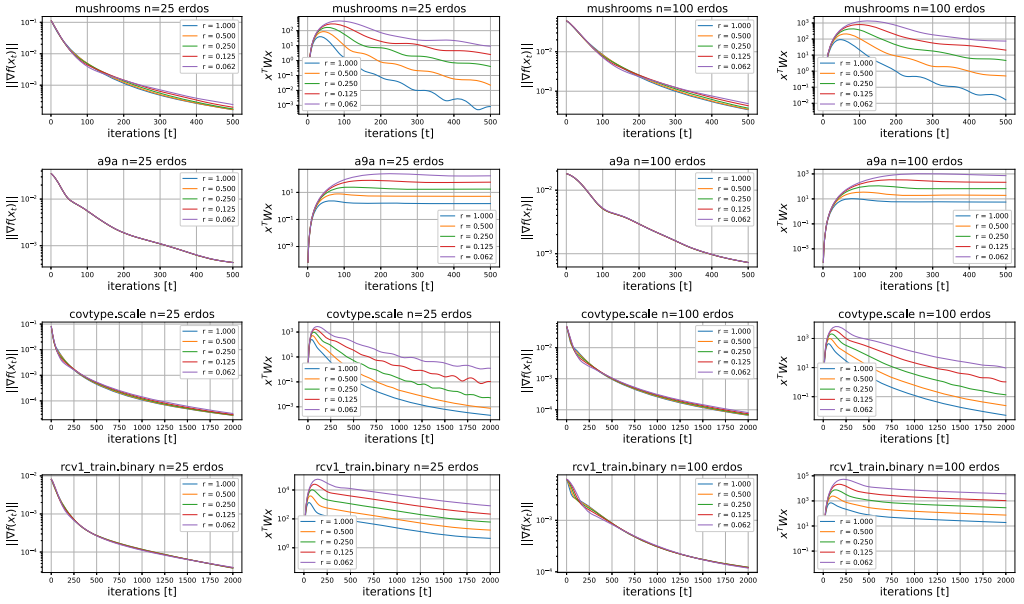

Fig. 4. Evolution of $\|\nabla f(x_t)\|$ and $x_t^T W x_t$ for various datasets, levels of λ and number of devices $n \in \{25, 100\}$ with erdos network.

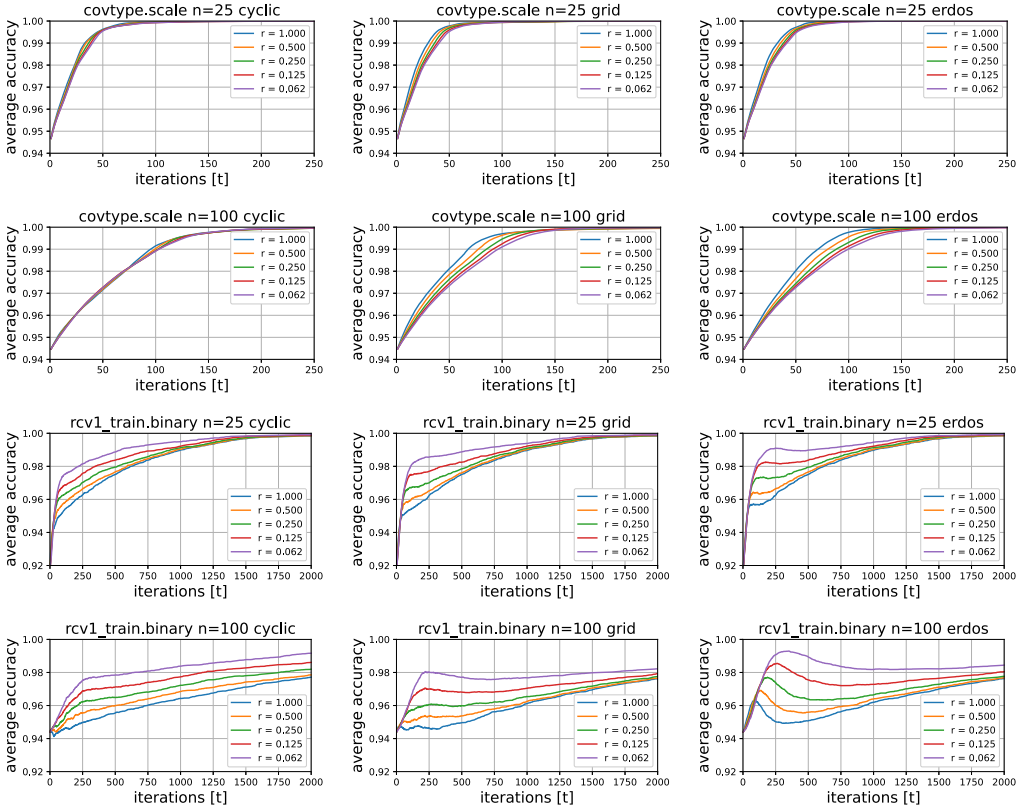

Fig. 5. Comparison of average accuracy of local models on local data for various communication networks and levels of λ (r).

local models. However, the *covtype* (more samples) and *rcv1* (more features) datasets were more challenging. For the *rcv1* dataset, we can see that initially (mainly due to the over-parametrization of the data) the local models achieve better accuracy (for smaller value of λ (r)), demonstrating the advantages of PFL.

5.4. Partial worker participation

One of the challenges of the FL setting is the fact that not all devices can always participate in all the communications [14,25]. To simulate such a scenario, we conducted the following two experiments:

1. **Randomly dropping communication edge(s).** For each iteration, and each communication edge e, the edge is kept with probability p_e, or dropped with probability $1-p_e$. The result is that the gossip matrix W is randomly modified at each iteration. In Figs. 6 and 7 we demonstrate empirically that keeping some communication edges with probability $p_e \in \{1.0, 0.9, 0.5, 0.1\}$ only mildly affects the convergence.

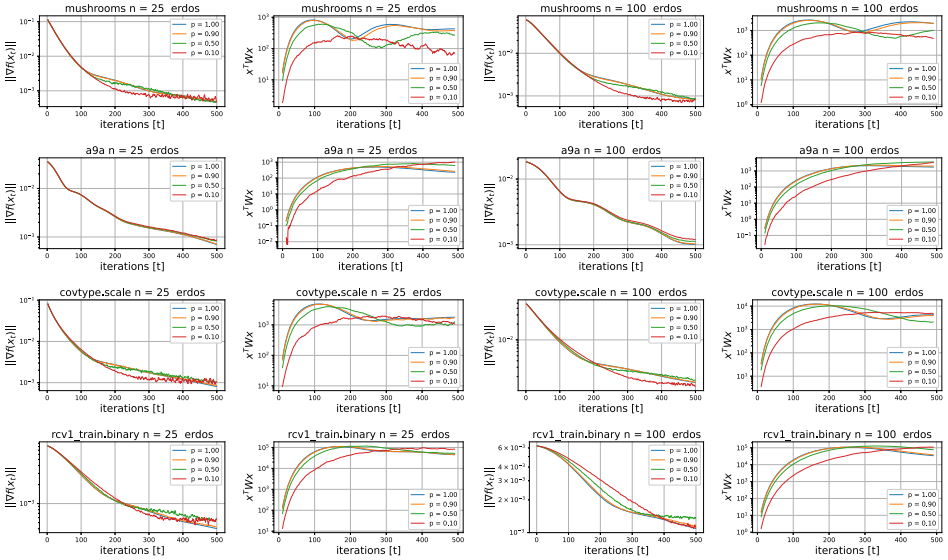

Fig. 6. Evolution of $\|\nabla f(x_t)\|$ and $x_t^T W x_t$ for various datasets, number of devices $n \in \{25, 100\}$ and different probability of keeping the communication edge p with the erdos network.

2. **Randomly dropping the device(s) from communication.** In this case, a subset of devices is randomly selected. In particular, at each iteration, a device is kept with probability p_d, and excluded/dropped with probability $1 - p_d$. As before, the effect is that the gossip matrix W is randomly modified at each iteration. In Figs. 8 and 9 we demonstrate empirically that keeping only some devices with probability $p_d \in \{1.0, 0.9, 0.5, 0.1\}$ only mildly affects the convergence.

5.5. The benefit of personalized training

In Section 5.2 we discussed the case when $\lambda \leq \frac{L}{\lambda_{\max}(W)}$ that allows for more personalization of local models. Note that, as discussed in Section 4.2, we use Algorithm 1 with different settings for $h_1(x)$ and $h_2(x)$ depending on the value of λ. In Fig. 10 we investigate the behavior of Algorithm 1 for $\lambda = r\frac{L}{\lambda_{\max}(W)}$ with $r \in \{0.125, 16\}$. Note that a larger value of λ (r) corresponds to larger penalization if the model deviates from the mean ($x^T W x$); consequently, this allows less personalization.

6. Conclusion

In this work we studied the problem of decentralized personalized federated learning. Problem (4) used a penalty term that was based upon the specific network structure, which was more appropriate than a 'deviation from the average' penalty in the decentralized setting. We presented lower bounds on the local communication and computation costs, and we presented algorithms that achieved these lower bounds. Numerical experiments demonstrated the benefits of this approach.

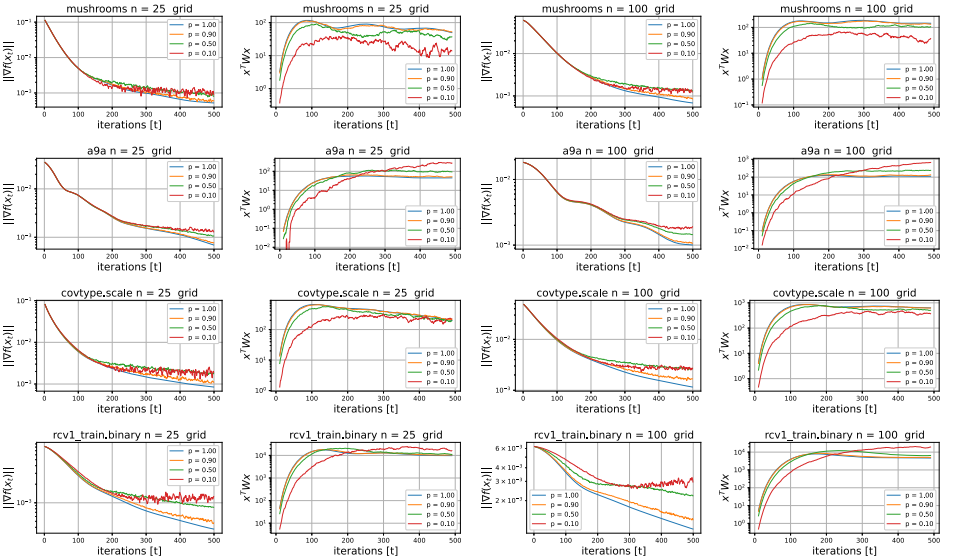

Fig. 7. Evolution of $\|\nabla f(x_t)\|$ and $x_t^T W x_t$ for various datasets, number of devices $n \in \{25, 100\}$ and different probability of keeping the communication edge p with grid network.

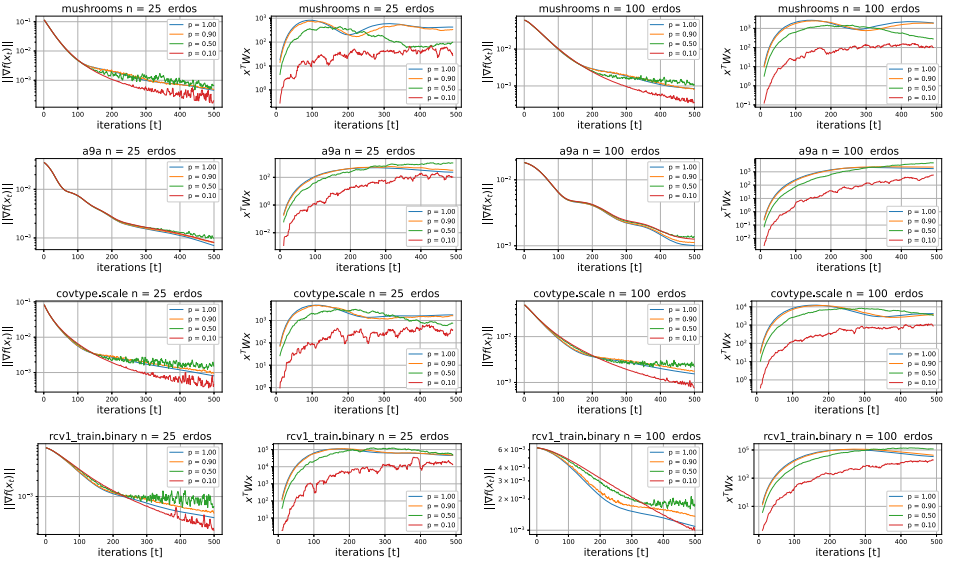

Fig. 8. Evolution of $\|\nabla f(x_t)\|$ and $x_t^T W x_t$ for various datasets, number of devices $n \in \{25, 100\}$ and different probability p of keeping the device in the communication with erdos network.

Interesting issues for further research are those related to the more practical features arising in a federated learning setup, including asynchronous and delayed transmissions, and compression of information to reduce communication cost, among others. It would also be interesting to perform numerical experiments using the Leaf framework (https://leaf.cmu.edu).

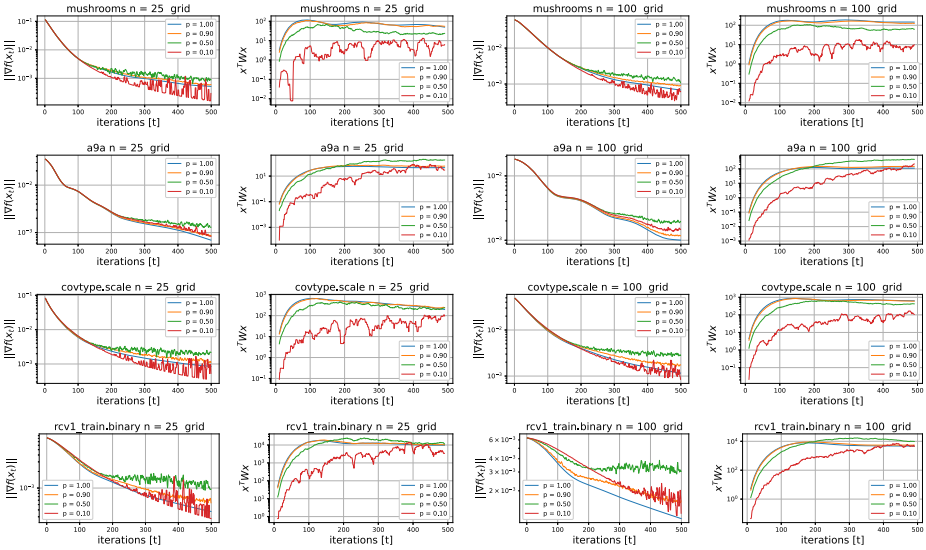

Fig. 9. Evolution of $\|\nabla f(x_t)\|$ and $x_t^T W x_t$ for various datasets, number of devices $n \in \{25, 100\}$ and different probability p of keeping the device in the communication with grid network.

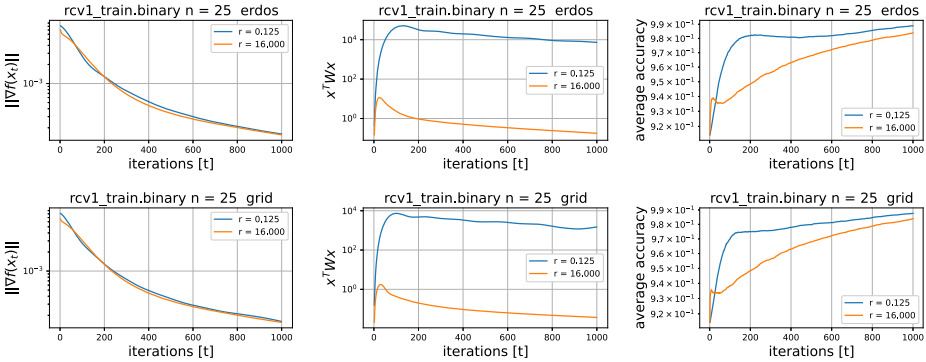

Fig. 10. Evolution of $\|\nabla f(x_t)\|$, $x_t^T W x_t$ and average accuracy of local models for rcv1 datasets and erdos and grid network. We compare with two levels of regularization: low with $\lambda = 0.125 \cdot \frac{L}{\lambda_{\max}(W)}$ and high with $\lambda = 16 \cdot \frac{L}{\lambda_{\max}(W)}$. Note that, although both regularization values give comparable $\|\nabla f(x_t)\|$, the average accuracy for the case with smaller penalization is better.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was partially conducted while A. Sadiev, A. Beznosikov, D. Dvinskikh were visiting research assistants and A. Gasnikov was a visiting scholar in Mohamed bin Zayed University of Artificial Intelligence (MBZUAI).

The work of E. Borodich was supported by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Moscow Institute of Physics and Technology dated November 1, 2021 No. 70-2021-00138.

Appendix A. Versions of Algorithms 1 and 3 for problem (4)

Algorithm 4 MA for $\lambda\lambda_{\max}(W) \geq L$.

Input: starting point $x_i^0 = x^0 \in \mathbb{R}^d$, no. of iterations K, parameter $\gamma = 2L > 0$, accuracy $\delta > 0$
Initialization: $A^0 = 0, \quad y_i^0 = x_i^0, \quad \tau = \frac{1}{2\gamma}$
for $k = 0, \ldots, K-1$ **do**
$a^{k+1} = \dfrac{\tau + \sqrt{\tau^2 + 4\tau A^k}}{2}$
$A^{k+1} = A^k + a^{k+1}$
Local update: $w_i^k = \dfrac{A^k}{A^{k+1}} y_i^k + \dfrac{a^{k+1}}{A^{k+1}} x_i^k$
Local computation: $u_i^k = \nabla f_i(w_i^k)$
Solve subproblem via gossip communications, i.e. find $\mathbf{y}^{k+1} \in \mathbb{R}^{nd}$, such that $\|\hat{\mathbf{y}}^{k+1} - \mathbf{y}^{k+1}\|_2^2 \leq \delta$, where

$$\hat{\mathbf{y}}^{k+1} = \underset{\mathbf{y} \in \mathbb{R}^{nd}}{\operatorname{argmin}} \left\{ \langle \nabla h_1(\mathbf{w}^k), \mathbf{y} - \mathbf{w}^k \rangle + h_2(\mathbf{y}) + \frac{\gamma}{2} \|\mathbf{y} - \mathbf{w}^k\|_2^2 \right\}$$

Compute z_i^k via gossip communication with neighbors: $\mathbf{z}^k = \lambda W \mathbf{y}^{k+1}$
Local update: $x_i^{k+1} = x_i^k - a^{k+1}(\nabla f_i(y_i^{k+1}) + z_i^k)$
end for
Output: $\{y_i^K\}$

Algorithm 5 MA for $\lambda\lambda_{\max}(W) < L$.

Input: starting point $x_i^0 = x^0 \in \mathbb{R}^d$, no. of iterations K, parameter $\gamma = 2L > 0$, accuracy $\delta > 0$
Initialization: $A^0 = 0, \quad y_i^0 = x_i^0, \quad \tau = \frac{1}{2\gamma}$
for $k = 0, \ldots, K-1$ **do**
$a^{k+1} = \dfrac{\tau + \sqrt{\tau^2 + 4\tau A^k}}{2}$
$A^{k+1} = A^k + a^{k+1}$
Local update: $w_i^k = \dfrac{A^k}{A^{k+1}} y_i^k + \dfrac{a^{k+1}}{A^{k+1}} x_i^k$
Compute u_i^k via gossip communication with neighbors: $\mathbf{u}^k = \lambda W \mathbf{w}^k$
Solve local subproblem, i.e. find $y_i^{k+1} \in \mathbb{R}^{nd}$, such that $\|\hat{y}_i^{k+1} - y_i^{k+1}\|_2^2 \leq \delta$, where

$$y_i^{k+1} = \underset{y_i \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \langle u_i^k, y_i - w_i^k \rangle + f_i(y_i) + \frac{\gamma}{2} \|y_i - w_i^k\|_2^2 \right\}$$

Compute z_i^k via gossip communication with neighbors: $\mathbf{z}^k = \lambda W \mathbf{y}^{k+1}$
Local update: $x_i^{k+1} = x_i^k - a^{k+1}(\nabla f_i(y_i^{k+1}) + z_i^k)$
end for
Output: $\{y_i^K\}$

Algorithm 6 ARDM.

Input: starting point $x_i^0 = x^0 \in \mathbb{R}^d$, number of iterations K, parameters $0 < \theta_1, \theta_2 < 1$, $\eta, \beta, \gamma > 0$, probabilities p, ρ

Initialization: $y_i^0 = z_i^0 = u_i^0 = x_i^0$ and $\hat{g}_i^0 = \lambda y_i^0 + \nabla f_i(y_i^0)$

for $k = 0, 1, 2, \dots K-1$ **do**
  Local update: $x_i^k = \theta_1 z_i^k + \theta_2 u_i^k + (1-\theta_1-\theta_2) y_i^k$
  Generate $\xi^k = \begin{cases} 1, & \text{with probability } 1-p \\ 0, & \text{with probability } p \end{cases}$
  if $\xi^k = 0$ **then**
    Compute b_i^k via gossip communication with neighbors: $\mathbf{b}^k = \frac{\lambda}{p} W \mathbf{x}^k$
    Local update: $g_i^k = b_i^k - a_i^k + \hat{g}_i^k$
  else
    Sample indices $m_1^k, \dots m_n^k$ for each node independently and uniformly from $[M]$
    Local computation: $g_i^k = \frac{1}{1-p}\left(\nabla f_{i,m_i^k}(x_i^k) - \nabla f_{i,m_i^k}(u_i^k)\right) + \hat{g}_i^k$
  end if
  Local update: $y_i^{k+1} = x_i^k - \eta g_i^k$
  Local update: $z_i^{k+1} = \beta z_i^k + (1-\beta) x_i^k + \frac{\gamma}{\eta}(y_i^{k+1} - x_i^k)$
  Generate $\xi^{k+\frac{1}{2}} = \begin{cases} 1, & \text{with prob. } 1-\rho \\ 0, & \text{with prob. } \rho \end{cases}$
  if $\xi^{k+1/2} = 0$ **then**
    $u_i^{k+1} = y_i^{k+1}$
    Compute a_i^{k+1} via gossip communication with neighbors: $\mathbf{a}^{k+1} = \frac{\lambda}{p} W \mathbf{u}^{k+1}$
    Compute c_i^k via gossip communication with neighbors: $\mathbf{c}^k = \lambda W \mathbf{y}^{k+1}$
    Local update: $\hat{g}_i^{k+1} = c_i^k + \nabla f_i(y_i^{k+1})$
  else
    $u_i^{k+1} = u_i^k$
    $a_i^{k+1} = a_i^k$
    $\hat{g}_i^{k+1} = \hat{g}_i^k$
  end if
end for

Appendix B. Proof of Theorem 1

In this section, we prove lower convergence bounds of algorithms satisfying Assumption 2 for the problem (4). To do this, we need to give an example of 'bad' functions that satisfy Assumption 1, and an example of a 'bad' arrangement of these functions in some graph with a 'bad' matrix $\hat{W}$ (Definition 1) with an upper bound of condition number χ. Following [20,8] we consider quadratic functions, and following [21], we construct a linear graph.

Let us start with the network. As the gossip matrix, we take the Laplacian of the linear graph. Then, for our problem (4), we get that the matrix W has the following form $W = \hat{W} \otimes I_d$, where $\hat{W} = \frac{1}{2}U$, and U is

$$U = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}. \tag{16}$$

It is known that the spectrum of the (scaled by $1/2$) Laplacian matrix of the linear graph with n vertices, is $2\sin^2\left(\frac{\pi k}{2n}\right)$ for $k = 0, \ldots, n-1$, [2]. Thus, the condition number is $\chi(n) = \frac{\sin^2\left(\frac{\pi(n-1)}{2n}\right)}{\sin^2\left(\frac{\pi}{2n}\right)}$. Since we consider $\chi \geq 3$, one can find $n \geq 3$ such that $\chi(n) \leq \chi < \chi(n+1)$. Moreover, for $n \geq 3$ we can guarantee that $\lambda_{\max}(n) \geq \frac{3}{2}$, $\frac{4}{n^2} \leq \lambda^+_{\min}(n) \leq \frac{5}{n^2}$ and $\chi(n+1) \leq \frac{1}{\sin^2\left(\frac{\pi}{2(n+1)}\right)} \leq \frac{(n+1)^2}{2}$. It turns out that if we choose as the 'bad' network, a linear graph with n vertices (where n is such that $\chi_n \leq \chi < \chi_{n+1}$), and take the Laplacian of this graph as the gossip matrix, then we satisfy Definition 1 and χ is an upper bound for the condition number of the gossip matrix. And one can note that $n - 1 > \sqrt{2\chi} - 2 \geq \frac{1}{5}\sqrt{\chi}$ (since $\chi \geq 3$), $1 \leq \frac{2}{3}\lambda_{\max}(n)$ and $\frac{4}{n^2} \leq \lambda^+_{\min}(n) \leq \frac{5}{n^2}$.

Now let us move on to the 'bad' functions. We choose the dimension of these functions equivalent to $d = 2T$ with large enough T (to be defined later). Next, we divide the nodes of the network into three types: the first type includes $\mathcal{V}_1 = \{1\}$, the second type includes $\mathcal{V}_2 = \{2, n-1\}$, the third type includes $\mathcal{V}_3 = \{n\}$. Each type of node has its own functions:

$$f_i(x) = \begin{cases} \frac{\mu}{2}\|x\|^2 + ax^{(1)} + \frac{c\lambda}{2}\left(\sum_{t=1}^{T-1}\left(x^{(2t)} - x^{(2t+1)}\right)^2\right) + \frac{b\lambda}{2}\left(x^{(2T)}\right)^2, & \text{if } i \in \mathcal{V}_1, \\ \phi \cdot \frac{\mu}{2}\|x\|^2, & \text{if } i \in \mathcal{V}_2, \\ \frac{\mu}{2}\|x\|^2 + \frac{c\lambda}{2}\left(\sum_{t=0}^{T-1}\left(x^{(2t+1)} - x^{(2t+2)}\right)^2\right), & \text{if } i \in \mathcal{V}_3, \end{cases} \tag{17}$$

where constants a, b, c will be defined shortly. The parameter ϕ takes two values: 1 or 0. We will consider both values below, we need 0 to simplify the mathematical calculations, note that in this case we slightly change the class of problems, since not all functions f_i are strongly convex and we slightly go beyond Assumption 1.

In the proof we will rely on [8]. In particular, we will prove similar (but not analogous) lemmas.

Let us introduce the solution of the problem (4) with (17). For the first type of node, we denote the solution by x^*, for the third type node by z^*, and for the second type nodes by $y_2^*, \ldots, y_{n-1}^*$. Using this notation we write down the optimality conditions for (4). First write down for x^*:

$$\left(\frac{\mu}{\lambda} + \frac{1}{2}\right)(x^*)^{(1)} + \frac{a}{\lambda} - \frac{1}{2}(y_2^*)^{(1)} = 0, \tag{18}$$

$$\left(c + \frac{\mu}{\lambda} + \frac{1}{2}\right)(x^*)^{(2t)} - c(x^*)^{(2t+1)} - \frac{1}{2}(y_2^*)^{(2t)} = 0, \quad \text{for } 1 \leq t \leq T-1, \tag{19}$$

$$\left(c + \frac{\mu}{\lambda} + \frac{1}{2}\right)(x^*)^{(2t+1)} - c(x^*)^{(2t)} - \frac{1}{2}(y_2^*)^{(2t+1)} = 0, \quad \text{for } 1 \leq t \leq T-1, \tag{20}$$

$$\left(\frac{\mu}{\lambda} + b + \frac{1}{2}\right)(x^*)^{(2T)} - \frac{1}{2}(y_2^*)^{(2T)} = 0. \tag{21}$$

Then for z^*:

$$\left(c+\frac{\mu}{\lambda}+\frac{1}{2}\right)(z^*)^{(2t-1)}-c(z^*)^{(2t)}-\frac{1}{2}(y^*_{n-1})^{(2t-1)}=0,\quad \text{for } 1\le t\le T, \tag{22}$$

$$\left(c+\frac{\mu}{\lambda}+\frac{1}{2}\right)(z^*)^{(2t)}-c(z^*)^{(2t-1)}-\frac{1}{2}(y^*_{n-1})^{(2t)}=0,\quad \text{for } 1\le t\le T, \tag{23}$$

Finally for $y_2^*,\dots,y_{n-1}^*$:

$$\left(1+\frac{\phi\mu}{\lambda}\right)(y_2^*)^{(t)}-\frac{1}{2}(y_3^*)^{(t)}-\frac{1}{2}(x^*)^{(t)}=0,\quad \text{for } 1\le t\le 2T, \tag{24}$$

$$\left(1+\frac{\phi\mu}{\lambda}\right)(y_i^*)^{(t)}-\frac{1}{2}(y_{i+1}^*)^{(t)}-\frac{1}{2}(y_{i-1}^*)^{(t)}=0,\quad \text{for } 1\le t\le 2T, \tag{25}$$

$$\left(1+\frac{\phi\mu}{\lambda}\right)(y_{n-1}^*)^{(t)}-\frac{1}{2}(y_{n-2}^*)^{(t)}-\frac{1}{2}(z^*)^{(t)}=0,\quad \text{for } 1\le t\le 2T. \tag{26}$$

First, we give a proof of the lemma that indicates a recursive connection of coordinates x^* and z^*. Before we introduce new notation:

$$w_t=\begin{cases}\begin{pmatrix}(z^*)^{(t)}\\(x^*)^{(t)}\end{pmatrix} & \text{if } t \text{ is even}\\ \begin{pmatrix}(x^*)^{(t)}\\(z^*)^{(t)}\end{pmatrix} & \text{if } t \text{ is odd}\end{cases}.$$

Lemma 1. *The sequence w_t satisfies the following recursion relation:*

$$w_{t+1}=Qw_t \quad \textit{with} \quad Q=\begin{pmatrix}-\frac{B}{2c} & \frac{1}{c}\left(c+\frac{\mu}{\lambda}+\frac{1}{2}-\frac{A}{2}\right)\\ -\frac{1}{c}\left(c+\frac{\mu}{\lambda}+\frac{1}{2}-\frac{A}{2}\right) & \frac{2}{Bc}\left(c+\frac{\mu}{\lambda}+\frac{1}{2}-\frac{A}{2}\right)^2-\frac{2c}{B}\end{pmatrix},$$

where

$$A=\left(1-\frac{1}{n-1}\right),\quad B=\frac{1}{n-1},\quad \textit{for}\quad \phi=0,$$

or

$$A=\frac{\omega_2^{n-2}-\omega_1^{n-2}}{\omega_2^{n-1}-\omega_1^{n-1}},\quad B=\frac{\omega_2-\omega_1}{\omega_2^{n-1}-\omega_1^{n-1}},\quad \textit{for}\quad \phi=1,$$

with $\omega_1=1+\frac{\mu}{\lambda}-\sqrt{\frac{2\mu}{\lambda}+\frac{\mu^2}{\lambda^2}}$ *and* $\omega_2=1+\frac{\mu}{\lambda}+\sqrt{\frac{2\mu}{\lambda}+\frac{\mu^2}{\lambda^2}}$.

Proof. We start from (24), (25), (26). One can note that we have recursion with two initial conditions:

$$(y_i^*)^{(t)}=\left(2+\frac{2\phi\mu}{\lambda}\right)(y_{i-1}^*)^{(t)}-(y_{i-2}^*)^{(t)}\quad \text{with}\quad (y_1^*)^{(t)}=(x^*)^{(t)},\ \ (y_n^*)^{(t)}=(z^*)^{(t)}.$$

If $\phi = 0$, the expressions for $(y_i^*)^{(t)}$ are as follows:

$$(y_i^*)^{(t)} = \left(\frac{i}{n-1} - \frac{1}{n-1}\right)(z^*)^{(t)} + \left(\frac{n}{n-1} - \frac{i}{n-1}\right)(x^*)^{(t)}.$$

In particular, $(y_2^*)^{(t)} = \left(1 - \frac{1}{n-1}\right)(x^*)^{(t)} + \frac{1}{n-1}(z^*)^{(t)}$ and $(y_{n-1}^*)^{(t)} = \left(1 - \frac{1}{n-1}\right)(z^*)^{(t)} + \frac{1}{n-1}(x^*)^{(t)}$. When $\phi = 1$, the expressions for $(y_i^*)^{(t)}$ become more complicated:

$$(y_i^*)^{(t)} = C_1\omega_1^{i-1} + C_2\omega_2^{i-1} = \frac{\omega_2^{n-i} - \omega_1^{n-i}}{\omega_2^{n-1} - \omega_1^{n-1}}(x^*)^{(t)} + \frac{\omega_2^{i-1} - \omega_1^{i-1}}{\omega_2^{n-1} - \omega_1^{n-1}}(z^*)^{(t)},$$

with $\omega_1 = 1 + \frac{\mu}{\lambda} - \sqrt{\frac{2\mu}{\lambda} + \frac{\mu^2}{\lambda^2}}$ and $\omega_2 = 1 + \frac{\mu}{\lambda} + \sqrt{\frac{2\mu}{\lambda} + \frac{\mu^2}{\lambda^2}}$. In particular, $(y_2^*)^{(t)} = \frac{\omega_2^{n-2} - \omega_1^{n-2}}{\omega_2^{n-1} - \omega_1^{n-1}}(x^*)^{(t)} + \frac{\omega_2 - \omega_1}{\omega_2^{n-1} - \omega_1^{n-1}}(z^*)^{(t)}$ and $(y_{n-1}^*)^{(t)} = \frac{\omega_2^{n-2} - \omega_1^{n-2}}{\omega_2^{n-1} - \omega_1^{n-1}}(z^*)^{(t)} + \frac{\omega_2 - \omega_1}{\omega_2^{n-1} - \omega_1^{n-1}}(x^*)^{(t)}$. In both cases of ϕ we have that $(y_2^*)^{(t)} = A \cdot (x^*)^{(t)} + B \cdot (z^*)^{(t)}$ and $(y_{n-1}^*)^{(t)} = A \cdot (z^*)^{(t)} + B \cdot (x^*)^{(t)}$ with some A and B. We can substitute these $(y_2^*)^{(t)}$ and $(y_{n-1}^*)^{(t)}$ into (19), (20), (22), (23) and have:

$$\begin{aligned}
&\left(c + \frac{\mu}{\lambda} + \frac{1}{2}\right)(x^*)^{(2t)} - c(x^*)^{(2t+1)} - \frac{A}{2}(x^*)^{(2t)} - \frac{B}{2}(z^*)^{(2t)} = 0, \quad \text{for } 1 \le t \le T-1,\\
&\left(c + \frac{\mu}{\lambda} + \frac{1}{2}\right)(x^*)^{(2t+1)} - c(x^*)^{(2t)} - \frac{A}{2}(x^*)^{(2t+1)} - \frac{B}{2}(z^*)^{(2t+1)} = 0, \quad \text{for } 1 \le t \le T-1,\\
&\left(c + \frac{\mu}{\lambda} + \frac{1}{2}\right)(z^*)^{(2t-1)} - c(z^*)^{(2t)} - \frac{A}{2}(z^*)^{(2t-1)} - \frac{B}{2} \cdot (x^*)^{(2t-1)} = 0, \quad \text{for } 1 \le t \le T,
\end{aligned} \tag{27}$$

$$\left(c + \frac{\mu}{\lambda} + \frac{1}{2}\right)(z^*)^{(2t)} - c(z^*)^{(2t-1)} - \frac{A}{2}(z^*)^{(2t)} - \frac{B}{2} \cdot (x^*)^{(2t)} = 0, \quad \text{for } 1 \le t \le T. \tag{28}$$

The first two expressions together can be rewritten as follows:

$$\begin{pmatrix} c & 0 \\ -c - \frac{\mu}{\lambda} - \frac{1}{2} + \frac{A}{2} & \frac{B}{2} \end{pmatrix} \begin{pmatrix} (x^*)^{(2t+1)} \\ (z^*)^{(2t+1)} \end{pmatrix} = \begin{pmatrix} c + \frac{\mu}{\lambda} + \frac{1}{2} - \frac{A}{2} & -\frac{B}{2} \\ -c & 0 \end{pmatrix} \begin{pmatrix} (x^*)^{(2t)} \\ (z^*)^{(2t)} \end{pmatrix},$$

or

$$\begin{aligned}
\begin{pmatrix} (x^*)^{(2t+1)} \\ (z^*)^{(2t+1)} \end{pmatrix} &= \begin{pmatrix} c & 0 \\ -c - \frac{\mu}{\lambda} - \frac{1}{2} + \frac{A}{2} & \frac{B}{2} \end{pmatrix}^{-1} \begin{pmatrix} c + \frac{\mu}{\lambda} + \frac{1}{2} - \frac{A}{2} & -\frac{B}{2} \\ -c & 0 \end{pmatrix} \begin{pmatrix} (x^*)^{(2t)} \\ (z^*)^{(2t)} \end{pmatrix} \\
&= \frac{2}{Bc} \begin{pmatrix} \frac{B}{2} & 0 \\ c + \frac{\mu}{\lambda} + \frac{1}{2} - \frac{A}{2} & c \end{pmatrix} \begin{pmatrix} c + \frac{\mu}{\lambda} + \frac{1}{2} - \frac{A}{2} & -\frac{B}{2} \\ -c & 0 \end{pmatrix} \begin{pmatrix} (x^*)^{(2t)} \\ (z^*)^{(2t)} \end{pmatrix} \\
&= \begin{pmatrix} \frac{1}{c}\left(c + \frac{\mu}{\lambda} + \frac{1}{2} - \frac{A}{2}\right) & -\frac{B}{2c} \\ \frac{2}{Bc}\left(c + \frac{\mu}{\lambda} + \frac{1}{2} - \frac{A}{2}\right)^2 - \frac{2c}{B} & -\frac{1}{c}\left(c + \frac{\mu}{\lambda} + \frac{1}{2} - \frac{A}{2}\right) \end{pmatrix} \begin{pmatrix} (x^*)^{(2t)} \\ (z^*)^{(2t)} \end{pmatrix}
\end{aligned}$$

$$= \begin{pmatrix} -\frac{B}{2c} & \frac{1}{c}\left(c+\frac{\mu}{\lambda}+\frac{1}{2}-\frac{A}{2}\right) \\ -\frac{1}{c}\left(c+\frac{\mu}{\lambda}+\frac{1}{2}-\frac{A}{2}\right) & \frac{2}{Bc}\left(c+\frac{\mu}{\lambda}+\frac{1}{2}-\frac{A}{2}\right)^2-\frac{2c}{B} \end{pmatrix} \begin{pmatrix} (z^*)^{(2t)} \\ (x^*)^{(2t)} \end{pmatrix}$$
$$= Q \begin{pmatrix} (z^*)^{(2t)} \\ (x^*)^{(2t)} \end{pmatrix}.$$

Similarly, from (27) and (28) one can get that

$$\begin{pmatrix} (z^*)^{(2t)} \\ (x^*)^{(2t)} \end{pmatrix} = Q \begin{pmatrix} (x^*)^{(2t-1)} \\ (z^*)^{(2t-1)} \end{pmatrix}.$$

Using the definition of w_t completes the proof. □

Then, we follow the idea from [8]. From the proof of the previous lemma we know that $(y_2^*)^{(t)} = A \cdot (x^*)^{(t)} + B \cdot (z^*)^{(t)}$. Then, substituting $(y_2^*)^{(1)}$ and $(y_2^*)^{(2T)}$ into (18) and (21), we obtain that the value of w_1 and w_{2T} depends on the parameters a and b. Hence, by varying the parameters a and b, one can obtain that $w_1, w_2, \ldots, w_{2T}$ are eigenvectors of the matrix Q, i.e. $w_2 = Qw_1 = \gamma w_1$ etc. This idea is implemented in the following lemma.

Lemma 2. *For any L, μ, and λ ($L \geq 2\mu$, and $\lambda\lambda^+_{\min}(W) \geq \mu$), there exists a choice of parameters a, b, c such that $w_1, w_2, \ldots w_{2T}$ are eigenvectors of matrix Q corresponding to the eigenvalue $\gamma \in (0;1)$, where*

$$\gamma \geq 1 - \max\left\{2\sqrt{\frac{\mu n^2}{\lambda}}, 3\sqrt{\frac{\mu}{L-\mu}}\right\}.$$

Moreover, the problem (4) + (17) with these parameters a, b, c satisfies Assumption 1.

Proof. First we give the values of a, b, and c:

$$c = \begin{cases} 1, & \text{for} \quad \mu + \lambda \leq L, \\ \frac{\mu}{\lambda} \cdot \delta = \frac{\mu}{\lambda} \cdot \frac{L-\mu}{\mu}, & \text{for} \quad \mu + \lambda > L, \end{cases}$$
$$b = \frac{B\alpha}{2} - \frac{\mu}{\lambda} - \frac{1}{2} + \frac{A}{2} \quad \text{and} \quad \text{any } a, \tag{29}$$

where

$$\alpha = -\frac{1-2A+A^2+B^2+4c-4Ac+4\frac{\mu}{\lambda}-4A\frac{\mu}{\lambda}+8c\frac{\mu}{\lambda}+4\frac{\mu^2}{\lambda^2}+\sqrt{\left(-1+2A-A^2+B^2-4\frac{\mu}{\lambda}+4A\frac{\mu}{\lambda}-4\frac{\mu^2}{\lambda^2}\right)\left(-1+2A-A^2+B^2-8c+8Ac-16c^2-4\frac{\mu}{\lambda}+4A\frac{\mu}{\lambda}-16c\frac{\mu}{\lambda}-4\frac{\mu^2}{\lambda^2}\right)}}{2B\left(-1+A-2c-2\frac{\mu}{\lambda}\right)}.$$

Let us check that the problem (4) + (17) satisfies Assumption 1. Note that by the choice of c, it suffices to verify that $0 \leq b\lambda \leq c\lambda \leq L - \mu$. We make this verification with Mathematica (here and below, when using Mathematica, we replace $\frac{\mu}{\lambda}$ with x). First, we check these inequalities when $\varphi = 0$ ($A = \frac{n-2}{n-1}$ and $B = \frac{1}{n-1}$):

- $b\lambda \leq c\lambda$ (or $\alpha \leq \frac{1}{B}\left(2c+1-A+2\frac{\mu}{\lambda}\right)$) for $0 < c \leq 1$ and $x = \frac{\mu}{\lambda} > 0$, $x = \frac{\mu}{\lambda} \leq \lambda^{+}_{\min} \leq \frac{5}{n^2}$ (since in Theorem 1 we assume that $\frac{\mu}{\lambda} \leq \lambda^{+}_{\min}$ and above we estimated that $\lambda^{+}_{\min} \leq \frac{5}{n^2}$)

```
In[1]:= FindInstance[-(1-2*A+A^2+B^2+4*c-4*A*c+4*x-4*A*x+8*c*x+4*x^2+Sqrt[(-1+2*A-A^2+B^2-4*x+4*A*x-4*x^2)*
          (-1+2*A-A^2+B^2-8c+8*A*c-16*c^2-4*x+4*A*x-16*c*x-4*x^2)])/(2*B*(-1+A-2*c-2*x))>(2*c+2*x+1-
          A)/B&&A==(n-2)/(n-1)&&B==1/(n-1)&&c>0&&c≤1&&x>0&&x*n^2≤5&&n≥3,{A,B,c,x,n}]
Out[1]= {}
```

- $0 \leq b$ (or $\alpha \geq \frac{1}{B}\left(1-A+2\frac{\mu}{\lambda}\right)$) for $0 < c \leq 1$ and $x = \frac{\mu}{\lambda} > 0$, $x = \frac{\mu}{\lambda} \leq \lambda^{+}_{\min} \leq \frac{5}{n^2}$

```
In[2]:= FindInstance[-(1-2*A+A^2+B^2+4*c-4*A*c+4*x-4*A*x+8*c*x+4*x^2+Sqrt[(-1+2*A-A^2+B^2-4*x+4*A*x-4*x^2)*
          (-1+2*A-A^2+B^2-8c+8*A*c-16*c^2-4*x+4*A*x-16*c*x-4*x^2)])/(2*B*(-1+A-2*c-2*x))<(2*x+1-A)/B&&
          A==(n-2)/(n-1)&&B==1/(n-1)&&c>0&&c≤1&&x>0&&x*n^2≤5&&n≥3,{A,B,c,x,n}]
Out[2]= {}
```

In the case of $\varphi = 1$, we replace the expressions for A and B from Lemma 1 by their Taylor approximations:

```
In[1]:= Series[((1+x+Sqrt[2*x+x^2])^(n-2)-(1+x-Sqrt[2*x+x^2])^(n-2))/((1+x+Sqrt[2*x+x^2])^(n-1)-(1+x-Sqrt[2*x+x^2])^(n-
          1)),{x,0,2}]
Out[1]= (-2+n)/(-1+n)+((-6+7n-2n^2)x)/(3(-1+n))+((-2+n)(3n-8n^2+4n^3)x^2)/(45(-1+n))+O[x]^(5/2)

In[2]:= Series[((1+x+Sqrt[2*x+x^2])-(1+x-Sqrt[2*x+x^2]))/((1+x+Sqrt[2*x+x^2])^(n-1)-(1+x-Sqrt[2*x+x^2])^(n-1)),{x,0,2}]
Out[2]= 1/(-1+n)+((2n-n^2)x)/(3(-1+n))+((-18n+37n^2-28n^3+7n^4)x^2)/(90(-1+n))+O[x]^(5/2)
```

$$
\begin{aligned}
A &\approx \frac{n-2}{n-1} - \frac{2n^2-7n+6}{3(n-1)}\frac{\mu}{\lambda} + \frac{(4n^3-8n^2+3n)(n-2)}{45(n-1)}\frac{\mu^2}{\lambda^2},\\
B &\approx \frac{n-2}{n-1} - \frac{n^2-2n}{3(n-1)}\frac{\mu}{\lambda} + \frac{7n^4-28n^3+37n^2-18n}{90(n-1)}\frac{\mu^2}{\lambda^2}.
\end{aligned}
\tag{30}
$$

Then, we can check inequalities for b:

- $b\lambda \leq c\lambda$ (or $\alpha \leq \frac{1}{B}\left(2c+1-A+2\frac{\mu}{\lambda}\right)$) for $0 < c \leq 1$ and $x = \frac{\mu}{\lambda} > 0$, $x = \frac{\mu}{\lambda} \leq \lambda^{+}_{\min} \leq \frac{5}{n^2}$

```
In[1]:= FindInstance[-(1-2*A+A^2+B^2+4*c-4*A*c+4*x-4*A*x+8*c*x+4*x^2+Sqrt[(-1+2*A-A^2+B^2-4*x+4*A*x-4*x^2)*
          (-1+2*A-A^2+B^2-8c+8*A*c-16*c^2-4*x+4*A*x-16*c*x-4*x^2)])/(2*B*(-1+A-2*c-2*x))>(2*c+2*x+1-
          A)/B&&A==(n-2)/(n-1)+(-6+7*n-2*n^2)*x/3/(n-1)+(n-2)*(3*n-8*n^2+4*n^3)*x^2/45/(n-1)&&B==1/(n-1)+(2*n-n^2)
          *x/3/(n-1)+(-18*n+37*n^2-28*n^3+7*n^4)*x^2/90/(n-1)&&c>0&&c≤1&&x>0&&x*n^2≤5&&n≥3,{A,B,c,x,n}]
Out[1]= {}
```

- $0 \leq b$ (or $\alpha \geq \frac{1}{B}\left(1-A+2\frac{\mu}{\lambda}\right)$) for $0 < c \leq 1$ and $x = \frac{\mu}{\lambda} > 0$, $x = \frac{\mu}{\lambda} \leq \lambda^{+}_{\min} \leq \frac{5}{n^2}$

```
In[2]:= FindInstance[-(1-2*A+A^2+B^2+4*c-4*A*c+4*x-4*A*x+8*c*x+4*x^2+Sqrt[(-1+2*A-A^2+B^2-4*x+4*A*x-4*x^2)*
          (-1+2*A-A^2+B^2-8c+8*A*c-16*c^2-4*x+4*A*x-16*c*x-4*x^2)])/(2*B*(-1+A-2*c-2*x))<(2*x+1-A)/B
          &&A==(n-2)/(n-1)+(-6+7*n-2*n^2)*x/3/(n-1)+(n-2)*(3*n-8*n^2+4*n^3)*x^2/45/(n-1)&&B==1/(n-1)+(2*n-n^2)*x/3
          /(n-1)+(-18*n+37*n^2-28*n^3+7*n^4)*x^2/90/(n-1)&&c>0&&c≤1&&x>0&&x*n^2≤5&&n≥3,{A,B,c,x,n}]
Out[2]= {}
```

```
In[1]:= Eigenvalues[{{-B / 2 / c , (c + x +1/2 - A / 2) / c}, {-(c + x +1/2 - A / 2) / c , 2 * (c + x +1/2 - A / 2)^2 / B / c - 2 * c / B}}]
```

$$\text{Out[1]}= \left\{\frac{4-8A+4A^2-4B^2+16c-16Ac+16x-16Ax+32cx+16x^2-\sqrt{-256B^2c^2+(-4+8A-4A^2+4B^2-16c+16Ac-16x+16Ax-32cx-16x^2)^2}}{16Bc},\right.$$
$$\left.\frac{4-8A+4A^2-4B^2+16c-16Ac+16x-16Ax+32cx+16x^2+\sqrt{-256B^2c^2+(-4+8A-4A^2+4B^2-16c+16Ac-16x+16Ax-32cx-16x^2)^2}}{16Bc}\right\}$$

```
In[2]:= Eigenvectors[{{-B / 2 / c , (c + x +1/2 - A / 2) / c}, {-(c + x +1/2 - A / 2) / c , 2 * (c + x +1/2 - A / 2)^2 / B / c - 2 * c / B}}]
```

$$\text{Out[2]}= \left\{\left\{-\frac{1-2A+A^2+B^2+4c-4Ac+4x-4Ax+8cx+4x^2+\sqrt{(-1+2A-A^2+B^2-4x+4Ax-4x^2)\,(-1+2A-A^2+B^2-8c+8Ac-16c^2-4x+4Ax-16cx-4x^2)}}{2B(-1+A-2c-2x)},\right.\right.$$
$$\left.\left.1\right\},\left\{-\frac{1-2A+A^2+B^2+4c-4Ac+4x-4Ax+8cx+4x^2-\sqrt{(-1+2A-A^2+B^2-4x+4Ax-4x^2)\,(-1+2A-A^2+B^2-8c+8Ac-16c^2-4x+4Ax-16cx-4x^2)}}{2B(-1+A-2c-2x)},\ 1\right\}\right\}$$

Next, we turn to eigenvalues and vectors. One can find them:

We take the smallest eigenvalue

$$\gamma=\frac{4-8A+4A^2-4B^2+16c-16Ac+16\frac{\mu}{\lambda}-16A\frac{\mu}{\lambda}+32c\frac{\mu}{\lambda}+16\frac{\mu^2}{\lambda^2}-\sqrt{-256B^2c^2+\left(-4+8A-4A^2+4B^2-16c+16Ac-16\frac{\mu}{\lambda}+16A\frac{\mu}{\lambda}-32c\frac{\mu}{\lambda}-16\frac{\mu^2}{\lambda^2}\right)^2}}{16Bc}$$

and the corresponding eigenvector

$$v=\begin{pmatrix}\alpha\\1\end{pmatrix}.$$

By simply substituting b from expression (29) and $(y_2^*)^{(t)}=A\cdot(x^*)^{(t)}+B\cdot(z^*)^{(t)}$ into equations (21), one can note that w_{2T} is an eigenvector of Q. It means that $\gamma w_{2T}=Qw_{2T}$ or $w_{2T}=\gamma Q^{-1}w_{2T}$. From Lemma 1 we also have $Q^{-1}w_{2T}=w_{2T-1}$. As the result, $w_{2T}=\gamma w_{2T-1}$, i.e. w_{2T-1} is also an eigenvector of Q. Continuing further, we can obtain that all vectors $w_{2T},\ldots,w_1$ are eigenvectors of Q. The choice of parameter a does not affect, it only determines the value of $\|w_1\|$.

Finally, we need to make sure that this γ satisfies the conditions of the lemma. Let us consider the three cases separately.

1) $\mu+\lambda\le L$. In this case $c=1$. We want to verify that $\gamma\in(0;1)$ and $\gamma\ge 1-2\sqrt{\frac{\mu n^2}{\lambda}}$. This inequality need to be checked with the constraints: $x=\frac{\mu}{\lambda}>0$, $x=\frac{\mu}{\lambda}\le\lambda_{\min}^{+}\le\frac{5}{n^2}$ (since in Theorem 1 we assume that $\frac{\mu}{\lambda}\le\lambda_{\min}^{+}$ and above we estimated that $\lambda_{\min}^{+}\le\frac{5}{n^2}$, when we construct the network). First, we check these inequalities when $\varphi=0$:

- $\gamma>0$

```
In[1]:= FindInstance[(4 - 8 * (n-2) / (n-1) + 4 * (n-2)^2 / (n-1)^2 - 4 / (n-1)^2 + 16 - 16 * (n-2) / (n-1) + 16 * x - 16 * x * (n-2) / (n-1) + 32 * x + 16 * x^2 -
        Sqrt[-256 / (n-1)^2 + (-4 + 8 * (n-2) / (n-1) - 4 * (n-2)^2 / (n-1)^2 + 4 / (n-1)^2 - 16 + 16 * (n-2) / (n-1) - 16 * x + 16 * x * (n-2) / (n-1) - 32 * x
          - 16 * x^2)^2]) / (16 / (n-1)) ≤ 0 && x > 0 && x * n^2 ≤ 5 && n ≥ 3, {x, n}]
Out[1]= {}
```

- $\gamma<1$

```
In[2]:= FindInstance[(4 - 8 * (n-2) / (n-1) + 4 * (n-2)^2 / (n-1)^2 - 4 / (n-1)^2 + 16 - 16 * (n-2) / (n-1) + 16 * x - 16 * x * (n-2) / (n-1) + 32 * x + 16 * x^2 -
        Sqrt[-256 / (n-1)^2 + (-4 + 8 * (n-2) / (n-1) - 4 * (n-2)^2 / (n-1)^2 + 4 / (n-1)^2 - 16 + 16 * (n-2) / (n-1) - 16 * x + 16 * x * (n-2) / (n-1) - 32 * x
          - 16 * x^2)^2]) / (16 / (n-1)) ≥ 1 && x > 0 && x * n^2 ≤ 5 && n ≥ 3, {x, n}]
Out[2]= {}
```

- $\gamma \geq 1 - 2\sqrt{\frac{\mu n^2}{\lambda}}$

```
In[3]:= FindInstance[(4 - 8 * (n-2) / (n-1) + 4 * (n-2)^2 / (n-1)^2 - 4 / (n-1)^2 + 16 - 16 * (n-2) / (n-1) + 16*x -16 * x * (n-2) / (n-1) + 32 * x + 16 * x^2 -
        Sqrt[-256 / (n-1)^2 + (-4 + 8 * (n-2) / (n-1) - 4 * (n-2)^2 / (n-1)^2 + 4 / (n-1)^2 - 16 + 16 * (n-2) / (n-1) - 16 * x + 16 * x * (n-2) / (n-1) - 32 * x
          - 16 * x^2)^2]) / (16 / (n-1)) - 1 + 2 * Sqrt[x * n^2] < 0 && x > 0 && x * n^2 ≤ 5 && n ≥ 3, {x, n}]
Out[3]= {}
```

In the case of $\varphi = 1$, we use (30):

- $\gamma > 0$

```
In[1]:= FindInstance[(4 - 8*((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) + 4 *((n-2) / (n-1) +
          (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1))^2 - 4 *(1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n
          +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))^2 + 16 - 16*((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 *
          n^3) * x^2 / 45 / (n-1)) + 16*x -16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) *
        x + 32 * x + 16 * x^2 - Sqrt[-256 * (1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))^2 + (-4 + 8*
          ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) - 4 * ((n-2) / (n-1) + (-6+7*n-2*n^
            2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1))^2 + 4 * (1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n +37 * n^2 -
            28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))^2 - 16 + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^
            2 / 45 / (n-1)) - 16 * x + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) * x - 32
          * x - 16 * x^2)^2]) / (16 * (1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))) ≤ 0 && x > 0 && x *
        n^2 ≤ 1 && n ≥ 3, {x, n}]
Out[1]= {}
```

- $\gamma < 1$

```
In[2]:= FindInstance[(4 - 8*((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) + 4 *((n-2) / (n-1) +
          (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1))^2 - 4 *(1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n
          +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))^2 + 16 - 16*((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 *
          n^3) * x^2 / 45 / (n-1)) + 16*x -16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) *
        x + 32 * x + 16 * x^2 - Sqrt[-256 * (1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))^2 + (-4 + 8*
          ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) - 4 * ((n-2) / (n-1) + (-6+7*n-2*n^
            2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1))^2 + 4 * (1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n +37 * n^2 -
            28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))^2 - 16 + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^
            2 / 45 / (n-1)) - 16 * x + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) * x - 32
          * x - 16 * x^2)^2]) / (16 * (1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))) ≥ 1 && x > 0 && x *
        n^2 ≤ 1 && n ≥ 3, {x, n}]
Out[2]= {}
```

- $\gamma \geq 1 - 2\sqrt{\frac{\mu n^2}{\lambda}}$

```
In[3]:= FindInstance[(4 - 8*((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) + 4 *((n-2) / (n-1) +
          (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1))^2 - 4 *(1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n
          +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))^2 + 16 - 16*((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 *
          n^3) * x^2 / 45 / (n-1)) + 16*x -16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) *
        x + 32 * x + 16 * x^2 - Sqrt[-256 * (1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))^2 + (-4 + 8*
          ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) - 4 * ((n-2) / (n-1) + (-6+7*n-2*n^
            2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1))^2 + 4 * (1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n +37 * n^2 -
            28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))^2 - 16 + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^
            2 / 45 / (n-1)) - 16 * x + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x / 3 / (n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 / 45 / (n-1)) * x - 32
          * x - 16 * x^2)^2]) / (16 * (1 / (n-1) + (2*n - n^2) * x / 3 / (n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 / 90 / (n-1))) - 1 + 2 * Sqrt[x
        * n^2] < 0 && x > 0 && x * n^2 ≤ 1 && n ≥ 3, {x, n}]
Out[3]= {}
```

2) $\mu + \lambda > L \geq \mu + \lambda\lambda_{\min}^{+}$. In this case $c = \frac{L-\mu}{\lambda} = \frac{\mu}{\lambda} \cdot \delta = x\delta$. We want to verify that $\gamma \in (0;1)$ and $\gamma \geq 1 - 2\sqrt{\frac{\mu n^2}{\lambda}}$. This inequality need to be checked with the constraints: $\delta \geq 1$ (since in Theorem 1 we assume that $L \geq 2\mu$ and then $\delta = \frac{L-\mu}{\mu} \geq 1$), $x = \frac{\mu}{\lambda} > 0$,

$x = \frac{\mu}{\lambda} \leq \lambda_{\min}^{+} \leq \frac{5}{n^2}$ and $\frac{1}{x} > \delta \geq \frac{1}{x} \cdot \lambda_{\min}^{+} \geq \frac{1}{x} \cdot \frac{4}{n^2}$ (constraints of the considered case). First, we check these inequalities when $\varphi = 0$:

- $\gamma > 0$

```
In[1]:= FindInstance[(4 - 8 * (n-2)/(n-1) + 4 * (n-2)^2/(n-1)^2 - 4 / (n-1)^2 + 16*(d * x) - 16* (d * x) * (n-2)/(n-1) + 16*x -16 * x * (n-2)/(n-1) + 32
          * (d * x) * x + 16 * x^2 - Sqrt[-256 / (n-1)^2 * (d * x)^2 + (-4 + 8* (n-2)/(n-1) - 4 * (n-2)^2/(n-1)^2 + 4 / (n-1)^2 - 16 * (d * x) + 16 * (d * x)
            * (n-2)/(n-1) - 16 * x + 16 * x * (n-2)/(n-1) - 32 * (d * x) * x - 16 * x^2)^2]) / (16 / (n-1) * (d * x)) ≤ 0 && d ≥ 1 && d * x < 1 && d * x * n^2
         ≥ 4 && x > 0 && x * n^2 ≤ 5 && n ≥ 3, {d, x, n}]
Out[1]= {}
```

- $\gamma < 1$

```
In[2]:= FindInstance[(4 - 8 * (n-2)/(n-1) + 4 * (n-2)^2/(n-1)^2 - 4 / (n-1)^2 + 16*(d * x) - 16* (d * x) * (n-2)/(n-1) + 16*x -16 * x * (n-2)/(n-1) + 32
          * (d * x) * x + 16 * x^2 - Sqrt[-256 / (n-1)^2 * (d * x)^2 + (-4 + 8* (n-2)/(n-1) - 4 * (n-2)^2/(n-1)^2 + 4 / (n-1)^2 - 16 * (d * x) + 16 * (d * x)
            * (n-2)/(n-1) - 16 * x + 16 * x * (n-2)/(n-1) - 32 * (d * x) * x - 16 * x^2)^2]) / (16 / (n-1) * (d * x)) ≥ 1 && d ≥ 1 && d * x < 1 && d * x * n^2
         ≥ 4 && x > 0 && x * n^2 ≤ 5 && n ≥ 3, {d, x, n}]
Out[2]= {}
```

- $\gamma \geq 1 - 2\sqrt{\frac{\mu n^2}{\lambda}}$

```
In[3]:= FindInstance[(4 - 8 * (n-2)/(n-1) + 4 * (n-2)^2/(n-1)^2 - 4 / (n-1)^2 + 16*(d * x) - 16* (d * x) * (n-2)/(n-1) + 16*x -16 * x * (n-2)/(n-1) + 32
          * (d * x) * x + 16 * x^2 - Sqrt[-256 / (n-1)^2 * (d * x)^2 + (-4 + 8* (n-2)/(n-1) - 4 * (n-2)^2/(n-1)^2 + 4 / (n-1)^2 - 16 * (d * x) + 16 * (d * x)
            * (n-2)/(n-1) - 16 * x + 16 * x * (n-2)/(n-1) - 32 * (d * x) * x - 16 * x^2)^2]) / (16 / (n-1) * (d * x)) - 1 + 2 * Sqrt[x * n^2] < 0 && d ≥ 1 && d
         * x < 1 && d * x * n^2 ≥ 4 && x > 0 && x * n^2 ≤ 5 && n ≥ 3, {d, x, n}]
Out[3]= {}
```

In the case of $\varphi = 1$, we use (30):

- $\gamma > 0$

```
In[1]:= FindInstance[(4 - 8*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1)) + 4 *((n-2) / (n-1) +
          (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1))^2 - 4 *(1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n
          +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/(n-1))^2 + 16 * (d * x) - 16*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 *
          n^2 + 4 * n^3) * x^2 /45/(n-1)) * (d * x) + 16*x -16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) *
          x^2 /45/(n-1)) * x + 32 * (d * x) * x + 16 * x^2 - Sqrt[-256 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) *
          x^2 /90/(n-1))^2 * (d * x)^2 + (-4 + 8* ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-
          1)) - 4 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1))^2 + 4 * (1 / (n-1) + (2*n -
          n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/(n-1))^2 - 16 * (d * x) + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x
          /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1)) * (d * x) - 16 * x + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) +
          (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1)) * x - 32 * (d * x) * x - 16 * x^2)^2]) / (16 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18
          * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/(n-1)) * (d * x)) ≤ 0 && d ≥ 1 && d * x < 1 && d * x * n^2 ≥ 4 && x > 0 && x * n^2 ≤ 5 && n ≥ 3, {d, x, n}]
Out[1]= {}
```

- $\gamma < 1$

```
In[2]:= FindInstance[(4 - 8*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1)) + 4 *((n-2) / (n-1) +
          (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1))^2 - 4 *(1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n
          +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/(n-1))^2 + 16 * (d * x) - 16*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 *
          n^2 + 4 * n^3) * x^2 /45/(n-1)) * (d * x) + 16*x -16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) *
          x^2 /45/(n-1)) * x + 32 * (d * x) * x + 16 * x^2 - Sqrt[-256 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) *
          x^2 /90/(n-1))^2 * (d * x)^2 + (-4 + 8* ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-
          1)) - 4 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1))^2 + 4 * (1 / (n-1) + (2*n -
          n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/(n-1))^2 - 16 * (d * x) + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x
          /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1)) * (d * x) - 16 * x + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) +
          (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1)) * x - 32 * (d * x) * x - 16 * x^2)^2]) / (16 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18
          * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/(n-1)) * (d * x)) ≥ 1 && d ≥ 1 && d * x < 1 && d * x * n^2 ≥ 4 && x > 0 && x * n^2 ≤ 5 && n ≥ 3, {d, x, n}]
Out[2]= {}
```

- $\gamma \geq 1 - 2\sqrt{\frac{\mu n^2}{\lambda}}$

```
In[3]:= FindInstance[(4 - 8 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1)) + 4 *((n-2) / (n-1) +
          (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1))^2 - 4 *(1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n
          +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/(n-1))^2 + 16 * (d * x) - 16*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 *
          n^2 + 4 * n^3) * x^2 /45/(n-1)) * (d * x) + 16*x -16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) *
          x^2 /45/(n-1)) * x + 32 * (d * x) * x + 16 * x^2 - Sqrt[-256 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) *
          x^2 /90/(n-1))^2 * (d * x)^2 + (-4 + 8* ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-
          1)) - 4 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1))^2 + 4 * (1 / (n-1) + (2*n -
          n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/(n-1))^2 - 16 * (d * x) + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x
          /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1)) * (d * x) - 16 * x + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) +
          (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/(n-1)) * x - 32 * (d * x) * x - 16 * x^2)^2]) / (16 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18
          * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/(n-1)) * (d * x)) - 1 + 2 * Sqrt[x * n^2] < 0 && d ≥ 1 && d * x < 1 && d * x * n^2 ≥ 4 && x > 0 && x * n^
        2 ≤ 5 && n ≥ 3, {d, x, n}]
Out[3]= {}
```

3) $\mu + \lambda\lambda^+_{\min} > L$. In this case $c = \frac{L-\mu}{\lambda} = \frac{1}{x} \cdot \delta = x\delta$. We want to verify that $\gamma \in (0; 1)$ and $\gamma \geq 1 - 3\sqrt{\frac{\mu}{L-\mu}}$. This inequality need to be checked with the constraints: $\delta \geq 1$, $x = \frac{1}{x} > 0$, $x = \frac{1}{x}\frac{1}{x} \leq \lambda^+_{\min} \leq \frac{5}{n^2}$ and $\delta < \frac{1}{x} \cdot \lambda^+_{\min} \leq \frac{1}{x} \cdot \frac{5}{n^2}$ (constraints of the considered case). First, we check these inequalities when $\varphi = 0$:

- $\gamma > 0$

```
In[1]:= FindInstance[(4 - 8 * (n-2) / (n-1) + 4 * (n-2)^2 / (n-1)^2 - 4 / (n-1)^2 + 16 * (d * x) - 16 * (d * x) * (n-2) / (n-1) + 16 * x - 16 * x * (n-2) / (n-1) + 32
          * (d * x) * x + 16 * x^2 - Sqrt[-256 / (n-1)^2 * (d * x)^2 + (-4 + 8 * (n-2) / (n-1) - 4 * (n-2)^2 / (n-1)^2 + 4 / (n-1)^2 - 16 * (d * x) + 16 * (d * x)
          * (n-2) / (n-1) - 16 * x + 16 * x * (n-2) / (n-1) - 32 * (d * x) * x - 16 * x^2)^2]) / (16 / (n-1) * (d * x)) ≤ 0 && d ≥ 1 && d * x * n^2 < 5 && x > 0 &&
        x * n^2 ≤ 5 && n ≥ 3, {d, x, n}]
Out[1]= {}
```

- $\gamma < 1$

```
In[2]:= FindInstance[(4 - 8 * (n-2) / (n-1) + 4 * (n-2)^2 / (n-1)^2 - 4 / (n-1)^2 + 16 * (d * x) - 16 * (d * x) * (n-2) / (n-1) + 16 * x - 16 * x * (n-2) / (n-1) + 32
          * (d * x) * x + 16 * x^2 - Sqrt[-256 / (n-1)^2 * (d * x)^2 + (-4 + 8 * (n-2) / (n-1) - 4 * (n-2)^2 / (n-1)^2 + 4 / (n-1)^2 - 16 * (d * x) + 16 * (d * x)
          * (n-2) / (n-1) - 16 * x + 16 * x * (n-2) / (n-1) - 32 * (d * x) * x - 16 * x^2)^2]) / (16 / (n-1) * (d * x)) ≥ 1 && d ≥ 1 && d * x * n^2 < 5 && x > 0 &&
        x * n^2 ≤ 5 && n ≥ 3, {d, x, n}]
Out[2]= {}
```

- $\gamma \geq 1 - 3\sqrt{\frac{\mu}{L-\mu}}$

```
In[3]:= FindInstance[(4 - 8 * (n-2) / (n-1) + 4 * (n-2)^2 / (n-1)^2 - 4 / (n-1)^2 + 16 * (d * x) - 16 * (d * x) * (n-2) / (n-1) + 16 * x - 16 * x * (n-2) / (n-1) + 32
          * (d * x) * x + 16 * x^2 - Sqrt[-256 / (n-1)^2 * (d * x)^2 + (-4 + 8 * (n-2) / (n-1) - 4 * (n-2)^2 / (n-1)^2 + 4 / (n-1)^2 - 16 * (d * x) + 16 * (d * x)
          * (n-2) / (n-1) - 16 * x + 16 * x * (n-2) / (n-1) - 32 * (d * x) * x - 16 * x^2)^2]) / (16 / (n-1) * (d * x)) - 1 + 3 * Sqrt[1/d] < 0 && d ≥ 1 && d * x
        * n^2 < 5 && x > 0 && x * n^2 ≤ 5 && n ≥ 3, {d, x, n}]
Out[3]= {}
```

In the case of $\varphi = 1$, we use (30):

- $\gamma > 0$

```
In[1]:= FindInstance[(4 - 8*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1)) + 4 *((n-2) / (n-1) +
        (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1))^2 - 4 *(1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n
         +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/ (n-1))^2 + 16 * (d * x) - 16*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 *
        n^2 + 4 * n^3) * x^2 /45/ (n-1)) * (d * x) + 16*x - 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) *
       x^2 /45/ (n-1)) * x + 32 * (d * x) * x + 16 * x^2 - Sqrt[-256 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) *
        x^2 /90/ (n-1))^2 * (d * x)^2 + (-4 + 8 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-
        1)) - 4 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1))^2 + 4 * (1 / (n-1) + (2*n -
        n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/ (n-1))^2 - 16 * (d * x) + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x
        /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1)) * (d * x) - 16 * x + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) +
        (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1)) * x - 32 * (d * x) * x - 16 * x^2)^2]) / (16 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18
        * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/ (n-1)) * (d * x)) ≤ 0 && d ≥ 1 && d * x * n^2 < 5 &&x > 0&&x * n^2 ≤ 1/5&&n ≥ 3, {d, x, n}]
Out[1]= {}
```

- $\gamma < 1$

```
In[2]:= FindInstance[(4 - 8*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1)) + 4 *((n-2) / (n-1) +
        (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1))^2 - 4 *(1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n
         +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/ (n-1))^2 + 16 * (d * x) - 16*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 *
        n^2 + 4 * n^3) * x^2 /45/ (n-1)) * (d * x) + 16*x - 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) *
       x^2 /45/ (n-1)) * x + 32 * (d * x) * x + 16 * x^2 - Sqrt[-256 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) *
        x^2 /90/ (n-1))^2 * (d * x)^2 + (-4 + 8 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-
        1)) - 4 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1))^2 + 4 * (1 / (n-1) + (2*n -
        n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/ (n-1))^2 - 16 * (d * x) + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x
        /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1)) * (d * x) - 16 * x + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) +
        (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1)) * x - 32 * (d * x) * x - 16 * x^2)^2]) / (16 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18
        * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/ (n-1)) * (d * x)) ≥ 1 && d ≥ 1 && d * x * n^2 < 5 &&x > 0&&x * n^2 ≤ 1/5&&n ≥ 3, {d, x, n}]
Out[2]= {}
```

- $\gamma \geq 1 - 3\sqrt{\frac{\mu}{L-\mu}}$

```
In[3]:= FindInstance[(4 - 8*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1)) + 4 *((n-2) / (n-1) +
        (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1))^2 - 4 *(1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n
         +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/ (n-1))^2 + 16 * (d * x) - 16*((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 *
        n^2 + 4 * n^3) * x^2 /45/ (n-1)) * (d * x) + 16*x - 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) *
       x^2 /45/ (n-1)) * x + 32 * (d * x) * x + 16 * x^2 - Sqrt[-256 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) *
        x^2 /90/ (n-1))^2 * (d * x)^2 + (-4 + 8 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-
        1)) - 4 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1))^2 + 4 * (1 / (n-1) + (2*n -
        n^2) * x /3 /(n-1) + (-18 * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/ (n-1))^2 - 16 * (d * x) + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x
        /3 /(n-1) + (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1)) * (d * x) - 16 * x + 16 * ((n-2) / (n-1) + (-6+7*n-2*n^2) * x /3 /(n-1) +
        (n-2) * (3 * n - 8 * n^2 + 4 * n^3) * x^2 /45/ (n-1)) * x - 32 * (d * x) * x - 16 * x^2)^2]) / (16 * (1 / (n-1) + (2*n - n^2) * x /3 /(n-1) + (-18
        * n +37 * n^2 - 28 * n^3 + 7 * n^4) * x^2 /90/ (n-1)) * (d * x)) - 1 + 3 * Sqrt[1/d] < 0 && d ≥ 1 && d * x * n^2 < 5 &&x > 0&&x * n^2 ≤ 1/5&&n ≥
    3, {d, x, n}]
Out[3]= {}
```

□

The previous Lemmas show what the solution of the problem (4) + (17) is. Now let us determine how quickly we can approach it.

Lemma 3. *Let the problem (4) + (17) be solved by any method that satisfies Assumption 2. Then after K iterations with q communication rounds, only the first $\left\lfloor \frac{q}{n-1} \right\rfloor$ coordinates of the global output can be non-zero while the rest of the $d - \left\lfloor \frac{q}{n-1} \right\rfloor$ coordinates are strictly equal to zero.*

Proof. We begin introducing some notation for our proof. Let

$$E_0 := \{0\}, \quad E_j := \operatorname{span}\{e_1, \ldots, e_j\}.$$

Note that, if we initialize all $x_i^0 = 0$, then we have $\mathcal{M}_{i,0} = E_0$.

Suppose that, at some given time k, for some j, $\mathcal{M}_{j,k} = E_l$. Let us analyze how $\mathcal{M}_{j,k}$ can change by performing only local computations.

We consider the case when l odd (case with even l can be analyzed the same way). After one local update, we have the following:

1) For node $j \in \mathcal{V}_1$, it holds

$$\mathcal{M}_{j,k+1} = E_l, \tag{31}$$

because of the block diagonal structure of (17). The situation does not change, no matter how many local computations one does.

2) For node $j \in \mathcal{V}_3$, it holds

$$\mathcal{M}_{j,k+1} = E_{l+1},$$

It means that, after local computations, one has an update in output and machine on $\mathcal{V}_3$ can progress by one new non-zero coordinate.

This means that we constantly have to transfer progress from the machine from $\mathcal{V}_1$ to the machine from $\mathcal{V}_3$ and back. Initially, all devices have zero coordinates. Further, the machine from $\mathcal{V}_1$ can receive the first nonzero coordinate (but only the first, the second is not), and the rest of the devices are left with all zeros. Next, we pass the first non-zero coordinate to the machine from $\mathcal{V}_3$. To do this, $n-1$ communication rounds are needed. By doing so, they can make the second coordinate non-zero, and then transfer this progress to the machine from $\mathcal{V}_1$. Then the process continues in the same way. This completes the proof. □

Now we are ready to complete the proof of Theorem 1. The previous reasoning, as well as Lemmas 1, 2, and 3, gives that we can construct the "bad" problem of type (4) with the "bad" network (satisfying Definition 1) as well as with the "bad" functions (17) (satisfying Assumption 1). Moreover, we know that only $\left\lfloor \frac{q}{n-1} \right\rfloor$ coordinates in the output can coincide with the solution, and the other coordinates are exactly zero. Then we just have to put $T = \frac{1}{2}\left(\max\{1, \log_\gamma \frac{1}{2}\} + \left\lfloor \frac{q}{n-1} \right\rfloor\right)$ in the dimension of the problem $d = 2T$, and obtain the following estimate on the outputs from $\mathcal{V}_1$ and $\mathcal{V}_3$:

$$\begin{aligned}
\frac{\|x^K - x^*\|^2 + \|z^K - z^*\|^2}{\|x^0 - x^*\|^2 + \|z^0 - z^*\|^2} &= \frac{\sum_{i=\left\lfloor \frac{q}{n-1} \right\rfloor + 1}^{2T} \|w_i\|^2}{\sum_{i=1}^{2T} \|w_i\|^2} \geq \frac{\sum_{i=\left\lfloor \frac{q}{n-1} \right\rfloor + 1}^{2T} \gamma^{i-1} \|w_1\|^2}{\sum_{i=1}^{2T} \gamma^{i-1} \|w_1\|^2} \\
&= \gamma^{\left\lfloor \frac{q}{n-1} \right\rfloor} \frac{\sum_{i=0}^{2T-1-\left\lfloor \frac{q}{n-1} \right\rfloor} \gamma^i}{\sum_{i=0}^{2T-1} \gamma^i} = \gamma^{\left\lfloor \frac{q}{n-1} \right\rfloor} \frac{1 - \gamma^{2T-\left\lfloor \frac{q}{n-1} \right\rfloor}}{1 - \gamma^{2T}} \\
&\geq \frac{1}{2} \gamma^{\left\lfloor \frac{q}{n-1} \right\rfloor} \geq \frac{1}{2} \left(1 - \max\left\{2\sqrt{\frac{\mu n^2}{\lambda}}, 3\sqrt{\frac{\mu}{L-\mu}}\right\}\right)^{\frac{q}{n-1}}.
\end{aligned}$$

In the other words it means that:

$$q = \Omega\left(\min\left\{\sqrt{\frac{\lambda(n-1)^2}{\mu n^2}}, \sqrt{\frac{(L-\mu)(n-1)^2}{\mu}}\right\}\log\frac{(\|x^0-x^*\|^2+\|z^0-z^*\|^2)}{\varepsilon}\right)$$
$$= \Omega\left(\min\left\{\sqrt{\frac{\lambda}{\mu}}, \sqrt{\frac{(L-\mu)(n-1)^2}{\mu}}\right\}\log\frac{(\|x^0-x^*\|^2+\|z^0-z^*\|^2)}{\varepsilon}\right)$$

When constructing the "bad" network, we proved that $n-1 > \sqrt{2\chi}-2 \geq \frac{1}{5}\sqrt{\chi}$ $1 \leq \frac{2}{3}\lambda_{\max}(n)$. Hence, we get

$$q = \Omega\left(\min\left\{\sqrt{\frac{\lambda(n-1)^2}{\mu n^2}}, \sqrt{\frac{(L-\mu)(n-1)^2}{\mu}}\right\}\log\frac{(\|x^0-x^*\|^2+\|z^0-z^*\|^2)}{\varepsilon}\right)$$
$$= \Omega\left(\min\left\{\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}, \sqrt{\frac{(L-\mu)\chi}{\mu}}\right\}\log\frac{(\|x^0-x^*\|^2+\|z^0-z^*\|^2)}{\varepsilon}\right).$$

Which is what we needed to prove. □

Appendix C. Proof of Theorem 4

For the following analysis, recall the auxiliary problem (7) from Algorithm 1 with $p=1$, $h_1(\mathbf{x})$ like sum component, $h_2(\mathbf{x})$ like $\frac{\lambda}{2}\langle \mathbf{x}, W\mathbf{x}\rangle$, which is restated for convenience:

$$\hat{\mathbf{y}}_{k+1} = \operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^{nd}}\left\{\langle \nabla h_1(\mathbf{w}_k), \mathbf{y}-\mathbf{w}_k\rangle + h_2(\mathbf{y}) + \frac{\gamma}{2}\|\mathbf{y}-\mathbf{w}_k\|_2^2\right\}$$

Now we look carefully at the auxiliary problem. This problem is $(L+\gamma)$-smooth and $(\mu+\gamma)$ strongly-convex, so we can apply L-Katyusha algorithm from [9]. The complexity of solving problem (7) is

$$\mathcal{O}\left(\left(M+\sqrt{\frac{M(\gamma+L)}{\gamma+\mu}}\right)\log\frac{1}{\delta}\right),$$

where δ denotes the accuracy of the solution to the auxiliary problem (7). The number of calls of the gradient of f is

$$N_{W\mathbf{x}} = \mathcal{O}\left(\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}\log\frac{1}{\varepsilon}\right) \tag{32}$$

while the number of calls of the gradient of $\mathcal{G}$ is

$$N_{\nabla f_k} = \mathcal{O}\left(\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}\left(M + \sqrt{\frac{M(L+\gamma)}{\mu+\gamma}}\right)\log\frac{1}{\varepsilon}\log\frac{1}{\delta}\right).$$

Taking γ be equal to $\lambda\lambda_{\max}(W)$, we get

$$N_{\nabla f_k} = \mathcal{O}\left(\left(M\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}} + \sqrt{\frac{ML}{\mu}}\right)\log\frac{1}{\varepsilon}\log\frac{1}{\delta}\right).$$

Now we consider δ, the accuracy of the auxiliary problem (7). According to Theorem 2, we can take δ as

$$\delta = \frac{\varepsilon\mu}{864^2(L+\lambda\lambda_{\max}(W)+\gamma)^2},$$

because the function $f(\mathbf{x})$ is L-smooth and $\frac{\lambda}{2}\langle\mathbf{x}, W\mathbf{x}\rangle$ is $\lambda\lambda_{\max}(W)$-smooth. □

Appendix D. Proof of Theorem 5

Let us use the additional notation $G(\mathbf{x}^k, \mathbf{u}^k) = \mathbf{g}^k + \lambda W\mathbf{u}^k + \nabla f(\mathbf{u}^k)$ for short. Let us consider our problem as a finite sum problem with $r+1$ terms.

$$F(\mathbf{x}) = \frac{1}{r}\sum_{j=1}^{r} g_j(\mathbf{x}) + g_{r+1}(\mathbf{x}),$$

where $g_j(\mathbf{x}) = \sum\limits_{k=1}^{n} f_{jk}(x_k)$ and $g_{r+1}(\mathbf{x}) = \frac{\lambda}{2}\langle\mathbf{x}, W\mathbf{x}\rangle$. For such a problem, one can use the results of the convergence of the variance reduction method L-Katyusha (Algorithm 3 from [9]) on which our method is based.

$$\begin{aligned}
&\mathbb{E}\left[\|\mathbf{g}^k - \nabla F(\mathbf{x}^k)\|^2\right] = \\
&= (1-p)\mathbb{E}\left[\left\|\frac{1}{1-p}\left(\nabla f_j(\mathbf{x}^k) - \nabla f_j(\mathbf{u}^k)\right) + \lambda W\mathbf{u}^k + \nabla f(\mathbf{u}^k) - \lambda W\mathbf{x}^k - \nabla f(\mathbf{x}^k)\right\|^2\right] \\
&+ p\mathbb{E}\left[\left\|\frac{\lambda}{p}\left(W\mathbf{x}^k - W\mathbf{u}^k\right) + \lambda W\mathbf{u}^k + \nabla f(\mathbf{u}^k) - \lambda W\mathbf{x}^k - \nabla f(\mathbf{x}^k)\right\|^2\right] \\
&= (1-p)\sum_{i=1}^{n}\sum_{j=1}^{r} p_j\left[\left\|\frac{1}{1-p}\left(\nabla f_{ij}(\mathbf{x}^k) - \nabla f_{ij}(\mathbf{u}^k)\right)\right.\right. \\
&\left.\left.+\lambda W\mathbf{u}^k + \nabla f_i(\mathbf{u}^k) - \lambda W\mathbf{x}^k - \nabla f_i(\mathbf{x}^k)\right\|^2\right] \\
&+ p\sum_{i=1}^{n}\left\|\frac{\lambda}{p}\left(W\mathbf{x}^k - W\mathbf{u}^k\right) + \lambda W\mathbf{u}^k + \nabla f_i(\mathbf{u}^k) - \lambda W\mathbf{x}^k - \nabla f_i(\mathbf{x}^k)\right\|^2
\end{aligned}$$

$$\leq (1-p)\sum_{i=1}^{n}\sum_{j=1}^{r} p_j \left\| \frac{1}{1-p}\left(\nabla f_{ij}(\mathbf{x}^k) - \nabla f_{ij}(\mathbf{u}^k)\right)\right\|^2 + p\left\|\frac{\lambda}{p}\left(W\mathbf{x}^k - W\mathbf{u}^k\right)\right\|^2$$

$$\leq \frac{1}{1-p}\sum_{i=1}^{n}\sum_{j=1}^{r} p_j \left\|\nabla f_{ij}(\mathbf{x}^k) - \nabla f_{ij}(\mathbf{u}^k)\right\|^2 + \frac{2\lambda\lambda_{\max}(W)}{p} D_{g_{r+1}}(\mathbf{u}^k, \mathbf{x}^k).$$

Choose $p_j = \frac{1}{r}$:

$$\begin{aligned}
&\mathbb{E}\left[\|\mathbf{g}^k - \nabla F(\mathbf{x}^k)\|^2\right] \\
&\quad\leq \frac{2L}{1-p}\sum_{j=1}^{n} D_{f_j}(\mathbf{u}^k, \mathbf{x}^k) + \frac{2\lambda\lambda_{\max}(W)}{p} D_{g_{r+1}}(\mathbf{u}^k, \mathbf{x}^k) \\
&\quad= \frac{2L}{1-p} D_f(\mathbf{u}^k, \mathbf{x}^k) + \frac{2\lambda\lambda_{\max}(W)}{p} D_{g_{r+1}}(\mathbf{u}^k, \mathbf{x}^k) \\
&\quad\leq \max\left\{\frac{2L}{1-p}, \frac{2\lambda\lambda_{\max}(W)}{p}\right\} D_F(\mathbf{u}^k, \mathbf{x}^k)
\end{aligned}$$

Choose $\mathcal{L} = \max\left\{\frac{L}{1-p}, \frac{\lambda\lambda_{\max}(W)}{p}\right\}$, then,

$$\mathbb{E}\left[\|G^k - \nabla F(\mathbf{x}^k)\|^2\right] \leq 2\mathcal{L} D_F(\mathbf{u}^k, \mathbf{x}^k).$$

Assumption 5.1 from [9] holds. By Proposition 5.1 from [9] iteration complexity of Algorithm 3 is

$$O\left(\left(\frac{1}{\rho} + \sqrt{\frac{L+\lambda\lambda_{\max}(W)}{\mu}} + \sqrt{\frac{\mathcal{L}}{\rho\mu}}\right)\log\frac{1}{\varepsilon}\right)$$

Note that optimal complexities in Algorithm 3 for local computations and communications are achieved on **different sets of p and ρ**. Let us get them separately.

* The local stochastic gradient complexity of a single iteration of Algorithm 3 is 0 if $\xi^k = 0$, $\xi^{k+\frac{1}{2}} = 1$, 1 if $\xi^k = 1$, $\xi^{k+\frac{1}{2}} = 1$, $r+1$ if $\xi^k = 1$, $\xi^{k+\frac{1}{2}} = 0$ and M if $\xi^k = 0$, $\xi^{k+\frac{1}{2}} = 0$.

$$\begin{aligned}
&\mathcal{O}\Bigg(((1-p)(1-\rho) + (M+1)(1-p)\rho + Mp\rho)\cdot \\
&\qquad\cdot\left(\frac{1}{\rho} + \sqrt{\frac{L+\lambda\lambda_{\max}(W)}{\mu}} + \sqrt{\frac{\mathcal{L}}{\rho\mu}}\right)\log\frac{1}{\varepsilon}\Bigg) \\
&\quad= \mathcal{O}\left((1-p+M\rho)\left(\frac{1}{\rho} + \sqrt{\frac{L+\lambda\lambda_{\max}(W)}{\mu}} + \sqrt{\frac{\mathcal{L}}{\rho\mu}}\right)\log\frac{1}{\varepsilon}\right).
\end{aligned}$$

For $\rho = \frac{1}{M}$, $p = \frac{\lambda\lambda_{\max}(W)}{L+\lambda\lambda_{\max}(W)}$ the total expected local stochastic gradient complexity of Algorithm 3 becomes

$$\begin{aligned}
&\mathcal{O}\left((1-p+M\rho)\left(\frac{1}{\rho}+\sqrt{\frac{L+\lambda\lambda_{\max}(W)}{\mu}}+\sqrt{\frac{\mathcal{L}}{\rho\mu}}\right)\log\frac{1}{\varepsilon}\right)\\
&\leq \mathcal{O}\left(2\left(M+\sqrt{\frac{L+\lambda\lambda_{\max}(W)}{\mu}}+\sqrt{\frac{M\mathcal{L}}{\mu}}\right)\log\frac{1}{\varepsilon}\right)\\
&= \mathcal{O}\left(\left(M+\sqrt{\frac{M(L+\lambda\lambda_{\max}(W))}{\mu}}\right)\log\frac{1}{\varepsilon}\right).
\end{aligned}$$

∗ The total communication complexity of Algorithm 3 is the sum of communication complexity coming from the full gradient computation (if statement that includes $\xi^{k+\frac{1}{2}}$) and the rest (if statement that includes ξ^k). The former requires a communication if $\xi^{k+\frac{1}{2}} = 0$, the latter if ξ^k is equal to 0. The expected total communication $\mathcal{O}(\rho+p)$ per iteration. Thus, the total communication complexity is bounded by

$$\mathcal{O}\left((p+\rho)\left(\frac{1}{\rho}+\sqrt{\frac{L+\lambda\lambda_{\max}(W)}{\mu}}+\sqrt{\frac{\mathcal{L}}{\rho\mu}}\right)\log\frac{1}{\varepsilon}\right).$$

For $\rho = p$, $p = \frac{\lambda\lambda_{\max}(W)}{L+\lambda\lambda_{\max}(W)}$ the total communication complexity of Algorithm 3 becomes

$$\begin{aligned}
&\mathcal{O}\left((\rho+p)\left(\frac{1}{\rho}+\sqrt{\frac{L+\lambda\lambda_{\max}(W)}{\mu}}+\sqrt{\frac{\mathcal{L}}{\rho\mu}}\right)\log\frac{1}{\varepsilon}\right)\\
&= \mathcal{O}\left(\left(1+\rho\sqrt{\frac{L+\lambda\lambda_{\max}(W)}{\mu}}+\sqrt{\frac{\rho(L+\lambda\lambda_{\max}(W))}{\mu}}\right)\log\frac{1}{\varepsilon}\right)\\
&= \mathcal{O}\left(\sqrt{\frac{\lambda\lambda_{\max}(W)(L+\lambda\lambda_{\max}(W))}{(L+\lambda\lambda_{\max}(W))\mu}}\log\frac{1}{\varepsilon}\right)\\
&= \mathcal{O}\left(\sqrt{\frac{\lambda\lambda_{\max}(W)}{\mu}}\log\frac{1}{\varepsilon}\right). \quad \square
\end{aligned}$$

References

[1] Zeyuan Allen-Zhu Katyusha, The first direct acceleration of stochastic gradient methods, J. Mach. Learn. Res. 18 (1) (2017) 8194–8244.
[2] William N. Anderson Jr, Thomas D. Morley, Eigenvalues of the laplacian of a graph, Linear Multilinear Algebra 18 (2) (1985) 141–145.
[3] Aleksandr Beznosikov, Vadim Sushko, Abdurakhmon Sadiev, Alexander Gasnikov, Decentralized personalized federated min-max problems, arXiv preprint, arXiv:2106.07289, 2021.

[4] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, Devavrat Shah, Randomized gossip algorithms, IEEE Trans. Inf. Theory 52 (6) (2006) 2508–2530.
[5] Chih-Chung Chang, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27, http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27 pp.
[6] Alexander Gasnikov, Darina Dvinskikh, Pavel Dvurechensky, Dmitry Kamzolov, Vladislav Matykhin, Dmitry Pasechnyk, Nazarii Tupitsa, Alexei Chernov, Accelerated meta-algorithm for convex optimization, arXiv preprint, arXiv:2004.08691, 2020.
[7] Eduard Gorbunov, Darina Dvinskikh, Alexander Gasnikov, Optimal decentralized distributed algorithms for stochastic convex optimization, arXiv preprint, arXiv:1911.07363, 2019.
[8] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, Peter Richtárik, Lower bounds and optimal algorithms for personalized federated learning, arXiv preprint, arXiv:2010.02372, 2020.
[9] Filip Hanzely, Dmitry Kovalev, Peter Richtarik, Variance reduced coordinate descent with acceleration: new method with a surprising application to finite-sum problems, arXiv preprint, arXiv:2002.04670, Feb 2020.
[10] Filip Hanzely, Peter Richtárik, Federated learning of a mixture of global and local models, arXiv preprint, arXiv:2002.05516, 2020.
[11] Filip Hanzely, Boxin Zhao, Mladen Kolar, Personalized federated learning: a unified framework and universal optimization techniques, 2021.
[12] Hadrien Hendrikx, Francis Bach, Laurent Massoulie, An optimal algorithm for decentralized finite sum optimization, arXiv preprint, arXiv:2005.10675, 2020.
[13] Rie Johnson, Tong Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 26, Curran Associates, Inc., 2013.
[14] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al., Advances and open problems in federated learning, Found. Trends Mach. Learn. 14 (1–2) (2021) 1–210.
[15] Jakub Konečnỳ, H. Brendan McMahan, Daniel Ramage, Peter Richtárik, Federated optimization: distributed machine learning for on-device intelligence, arXiv preprint, arXiv:1610.02527, 2016.
[16] Viraj Kulkarni, Milind Kulkarni, Aniruddha Pant, Survey of personalization techniques for federated learning, in: 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), IEEE, 2020, pp. 794–797.
[17] Huan Li, Cong Fang, Wotao Yin, Zhouchen Lin, Decentralized accelerated gradient methods with increasing penalty parameters, IEEE Trans. Signal Process. 68 (2020) 4855–4870.
[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Aguera y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[19] Angelia Nedic, Asuman Ozdaglar, Distributed subgradient methods for multi-agent optimization, IEEE Trans. Autom. Control 54 (1) (2009) 48–61.
[20] Yurii Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, vol. 87, Springer Science & Business Media, 2003.
[21] Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, Laurent Massoulié, Optimal algorithms for smooth and strongly convex distributed optimization in networks, arXiv preprint, arXiv:1702.08704, 2017.
[22] Shai Shalev-Shwartz, Shai Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.
[23] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, Ameet Talwalkar, Federated multi-task learning, arXiv preprint, arXiv:1705.10467, 2017.
[24] Vladislav Tominin, Yaroslav Tominin, Ekaterina Borodich, Dmitry Kovalev, Alexander Gasnikov, Pavel Dvurechensky, On accelerated methods for saddle-point problems with composite structure, arXiv preprint, arXiv:2103.09344, 2021.
[25] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H. Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al., A field guide to federated optimization, arXiv preprint, arXiv:2107.06917, 2021.
[26] Weiran Wang, Jialei Wang, Mladen Kolar, Nathan Srebro, Distributed stochastic multi-task learning with graph regularization, arXiv preprint, arXiv:1802.03830, 2018.