

---

# ADOM: Accelerated Decentralized Optimization Method for Time-Varying Networks

---

Dmitry Kovalev<sup>1</sup> Egor Shulgin<sup>1</sup> Peter Richtárik<sup>1</sup> Alexander Rogozin<sup>2,3</sup> Alexander Gasnikov<sup>2,3</sup>

## Abstract

We propose ADOM – an accelerated method for smooth and strongly convex decentralized optimization over time-varying networks. ADOM uses a dual oracle, i.e., we assume access to the gradient of the Fenchel conjugate of the individual loss functions. Up to a constant factor, which depends on the network structure only, its communication complexity is the same as that of accelerated Nesterov gradient method (Nesterov, 2003). To the best of our knowledge, only the algorithm of Rogozin et al. (2019) has a convergence rate with similar properties. However, their algorithm converges under the very restrictive assumption that the number of network changes can not be greater than a tiny percentage of the number of iterations. This assumption is hard to satisfy in practice, as the network topology changes usually can not be controlled. In contrast, ADOM merely requires the network to stay connected throughout time.

## 1. Introduction

We study the *decentralized optimization* problem

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

where each function  $f_i: \mathbb{R}^d \rightarrow \mathbb{R}$  is stored on a compute node  $i \in [n] := \{1, 2, \dots, n\}$ . We assume that the nodes are connected through a *communication network* defined by an undirected connected graph. Each node can perform computations based on its local state and data, and can directly communicate with its neighbors only. Further, we assume the functions  $f_i$  to be smooth and strongly convex. Such decentralized optimization problems have been studied heavily

---

<sup>1</sup>King Abdullah University of Science and Technology, Thuwal, Saudi Arabia <sup>2</sup>Moscow Institute of Physics and Technology, Dolgoprudny, Russia <sup>3</sup>Higher School of Economics, Moscow, Russia. Correspondence to: Dmitry Kovalev <dakovalev1@gmail.com>.

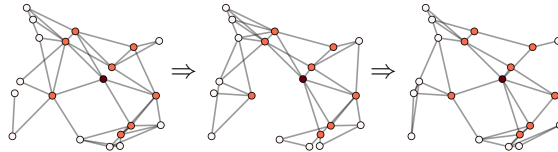


Figure 1. A sample time-varying network with  $n = 20$  nodes.

(Gorbunov et al., 2020b), and arise in many applications, including estimation by sensor networks (Rabbat & Nowak, 2004), network resource allocation (Beck et al., 2014), cooperative control (Giselsson et al., 2013), distributed spectrum sensing (Bazerque & Giannakis, 2009), power system control (Gan et al., 2012) and federated learning (Li et al., 2020a; Kovalev et al., 2021). When the network is not allowed to change in time, a lower communication complexity bound has been established by Scaman et al. (2017). This bound is tight as there is a matching upper bound both in the case when a *dual oracle* is assumed (Scaman et al., 2017), which means that we have access to the gradient of the Fenchel conjugate of the functions  $f_i(x)$ , and also in the case when a *primal oracle* is assumed (Kovalev et al., 2020b), which means that we have access to the gradient of the functions  $f_i(x)$  themselves.

### 1.1. Time-varying networks

In this work, we study the situation when the links in the communication network are allowed to change over time (for an illustration, see Figure 1). Such *time-varying networks* (Zadeh, 1961; Kolar et al., 2010) are ubiquitous in many complex systems and practical applications. In sensor networks, for example, changes in the link structure occur when the sensors are in motion, and due to other disturbances in the wireless signal connecting pairs of nodes. We envisage that a similar regime will be supported in future-generation federated learning systems (Konečný et al., 2016; McMahan et al., 2017; Kovalev et al., 2021), where the communication pattern among pairs of mobile devices or mobile devices and edge servers will be dictated by their physical proximity, which naturally changes over time. Our work can be partially understood as an attempt to contribute to the algorithmic foundations of this nascent field.

## 1.2. On the smooth and strongly convex regime

As mentioned earlier, throughout this paper we restrict each function  $f_i(x)$  to be  $L$ -smooth and  $\mu$ -strongly convex. That is, we require that the inequalities  $f_i(x) \geq f_i(y) + \langle \nabla f_i(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2$  and  $f_i(x) \leq f_i(y) + \langle \nabla f_i(y), x - y \rangle + \frac{L}{2} \|x - y\|^2$  hold for all nodes  $i \in [n]$  and all  $x, y \in \mathbb{R}^d$ . This naturally leads to the quantity

$$\kappa := L/\mu \quad (2)$$

known as the *condition number* of function  $f$ . As we shall see, current understanding of decentralized optimization over time-varying networks is insufficient even in this setting, and we believe that the key technical issues we face at present do not come from the difficulty of the function class, but from the algorithmic and modeling aspect of dealing with the decentralized and time-varying nature of the problem. Thus, focusing on smooth and strongly convex problems should not be seen as a weakness, but as a necessary step in the quest to make a significant advance in our understanding of how efficient decentralized methods should be designed in the time-varying network regime.

## 1.3. Methods for time-varying networks

To the best of our knowledge, there is only a handful of algorithms for solving the decentralized optimization problem (1) that enjoy a *linear convergence rate* in the time-varying regime under smoothness and strong convexity assumptions. These include DIGing (Nedic et al., 2017) and Push-Pull Gradient Method (Pu et al., 2020), which use the primal oracle, and PANDA (Maros & Jaldén, 2018), which uses the dual oracle. While linear, their rates are slow in comparison to the best methods “on the market” at present (see Table 1).

A well known mechanism for improving the convergence rates of standard gradient type methods is to apply or adapt Nesterov *acceleration* (Nesterov, 2003), whose goal is to reduce the dependence of the method on the condition number  $\kappa$  associated with the problem, the condition number  $\chi$  associated with the network structure (see (14) in Section 4.3), or both. However, doing this is nontrivial in the decentralized time-varying setting.

## 2. Summary of Contributions

We now briefly outline the main contributions:

### 2.1. New algorithm

In this paper we propose an accelerated algorithm—ADOM (Algorithm 2)—for smooth and strongly convex decentralized optimization over time-varying networks. This algorithm uses the dual oracle, and is based on a careful generalization of the Projected Nesterov Gradient Descent method (PNGD; Algorithm 1).

### 2.2. Convergence analysis

We prove that ADOM enjoys the rate  $\mathcal{O}(\chi\kappa^{1/2} \log \frac{1}{\varepsilon})$  (see Thm 1), which matches the  $\mathcal{O}(\kappa^{1/2} \log \frac{1}{\varepsilon})$  rate of PNGD in the special case of a fully connected time-invariant network.

### 2.3. Innovations in the analysis

Our analysis requires several new insights and tools. First, we rely on the *new observation* that *decentralized communication can be seen as the application of a certain contractive compression operator* (see Section 4.4). This operator is linear, but may be *biased*, which raises significant challenges. While the use of *unbiased* compression operators, such as sparsification and quantization, is increasingly popular in modern literature on distributed optimization in the parameter server framework<sup>1</sup>, we only know of a handful of results combining compression with acceleration (Li et al., 2020b; Qian et al., 2020). Of these, the first handles *unbiased* compressors only, and the second is the only work we know of successfully combining biased communication compression and acceleration. However, their work makes use of a different acceleration mechanism from ours, and it is not clear how to extend it to decentralized optimization. We are not aware of any results combining the use of biased compressors, acceleration and decentralized communication, even if we allow for the networks to be time-invariant. The observation that decentralized communication can be modeled as the application of a certain contractive compressor allows us to design a bespoke *error-feedback* mechanism, previously studied in other settings by Stich & Karimireddy (2019); Karimireddy et al. (2019); Beznosikov et al. (2020); Gorbunov et al. (2020a), for achieving acceleration despite dealing with a biased compressor.

### 2.4. Comparison to accelerated methods designed for time-varying networks

While there were attempts to design accelerated algorithms that could deal with time-varying networks, only several methods provide sub-quadratic dependence on  $\chi$ : Acc-DNGD (Qu & Li, 2019), Mudag (Ye et al., 2020), and the Accelerated Penalty Method (APM) (Rogozin et al., 2020; Li et al., 2018). Acc-DNGD has  $\mathcal{O}(\chi^{3/2})$  dependence on  $\chi$ , which is worse than the linear dependence on  $\chi$  shared by Mudag, APM and our method ADOM. Moreover, Acc-DNGD has  $\mathcal{O}(\kappa^{5/7})$  dependence on  $\kappa$  and Mudag has  $\mathcal{O}(\kappa^{1/2} \log \kappa)$  dependence, which is worse than the  $\mathcal{O}(\kappa^{1/2})$  dependence of APM and our method ADOM. Lastly, APM has a square-logarithmic dependence on  $1/\varepsilon$ , which is worse than the dependence of all the other methods on this quantity. These results are summarized in Table 1. In summary,

<sup>1</sup>Distributed optimization in a parameter server framework is mathematically equivalent to the setting where communication happens over a fully connected time-invariant network.

Table 1. A review of decentralized optimization algorithms capable of working in the time-varying network regime, with guarantees. Complexity terms **highlighted in red** represent the best known dependencies. Our method is the only method with best known dependencies in all terms.

Algorithm	Communication complexity
DIGing Nedic et al. (2017)	$\mathcal{O}\left(n^{1/2}\chi^2\kappa^{3/2}\log\frac{1}{\epsilon}\right)$
PANDA Maros & Jaldén (2018)	$\mathcal{O}\left(\chi^2\kappa^{3/2}\log\frac{1}{\epsilon}\right)$
Acc-DNGD Qu & Li (2019)	$\mathcal{O}\left(\chi^{3/2}\kappa^{5/7}\log\frac{1}{\epsilon}\right)$
APM Li et al. (2018)	$\mathcal{O}\left(\chi\kappa^{1/2}\log^2\frac{1}{\epsilon}\right)$
Mudag Ye et al. (2020)	$\mathcal{O}\left(\chi\kappa^{1/2}\log(\kappa)\log\frac{1}{\epsilon}\right)$
<b>ADOM (Algorithm 2) THIS PAPER</b>	$\mathcal{O}\left(\chi\kappa^{1/2}\log\frac{1}{\epsilon}\right)$

ADOM achieves the new state-of-the-art rate for decentralized optimization over time-varying networks.

### 2.5. Comparison to DNM of Rogozin et al. (2019)

We left one relevant method out from the above comparison – the Distributed Nesterov Method (DNM) of Rogozin et al. (2019). This method has  $\mathcal{O}(\chi^{1/2})$  dependence on  $\chi$ . However, DNM converges under the very restrictive assumption requiring the *number of network changes to not exceed a tiny percentage of the number of iterations*. This assumption is hard to satisfy in practice, as the changes in the network topology usually can not be controlled and happen independently of the algorithm run. In contrast, our algorithm just requires the network to be connected all the time. In Figure 2 we give a representative comparison of the workings of our method ADOM and DNM in a regime where the number of network changes exceeds the theoretical limit. While ADOM converges, DNM often diverges, which shows that DNM is not robust to the network dynamics, and that the restrictive assumption is crucial to their analysis.

## 3. Problem Formulation and Projected Nesterov Gradient Descent

The design of our method is based on a particular reformulation of problem (1), which we now describe.

### 3.1. Reformulation via Lifting

Consider function  $F: (\mathbb{R}^d)^\mathcal{V} \rightarrow \mathbb{R}$  defined by

$$F(x) = \sum_{i \in \mathcal{V}} f_i(x_i), \quad (3)$$

where  $x = (x_1, \dots, x_n) \in (\mathbb{R}^d)^\mathcal{V}$  and  $\mathcal{V} := [n]$  denotes the set of compute nodes. Then  $F$  is  $L$ -smooth  $\mu$ -strongly

convex since the individual functions  $f_i$  are. Consider also the so called *consensus space*  $\mathcal{L} \subset (\mathbb{R}^d)^\mathcal{V}$  defined by

$$\mathcal{L} := \{(x_1, \dots, x_n) \in (\mathbb{R}^d)^\mathcal{V} : x_1 = \dots = x_n\}. \quad (4)$$

Using this notation, we arrive at an equivalent formulation of problem (1), which we call the *primal formulation*:

$$\min_{x \in \mathcal{L}} F(x). \quad (5)$$

Since the function  $F(x)$  is strongly convex, this problem has a unique solution, which we denote as  $x^* \in \mathcal{L}$ .

### 3.2. Dual Problem

It is a well known fact that problem (5) has an equivalent *dual formulation* of the form

$$\min_{z \in \mathcal{L}^\perp} F^*(z), \quad (6)$$

where  $F^*$  is the Fenchel transform of  $F$  and  $\mathcal{L}^\perp \subset (\mathbb{R}^d)^\mathcal{V}$  is the orthogonal complement to the space  $\mathcal{L}$ , given as follows:

$$\mathcal{L}^\perp = \{(z_1, \dots, z_n) \in (\mathbb{R}^d)^\mathcal{V} : \sum_{i=1}^n z_i = 0\}. \quad (7)$$

Note that the function  $F^*(z)$  is  $\frac{1}{\mu}$ -smooth and  $\frac{1}{L}$ -strongly convex (Rockafellar, 1970). Hence, problem (6) also has a unique solution, which we denote as  $z^* \in \mathcal{L}^\perp$ .

### 3.3. Projected Nesterov Gradient Descent

A natural way to tackle problem (6) is to use a projected version of Nesterov’s accelerated gradient method: Projected Nesterov Gradient Descent (PNGD) (Nesterov, 2003). This algorithm requires us to calculate projection onto the set  $\mathcal{L}^\perp$ , which can be written in the closed form

$$\text{proj}_{\mathcal{L}^\perp}(g) := \arg \min_{z \in \mathcal{L}^\perp} \|g - z\|^2 = \mathbf{P}g, \quad (8)$$

where  $g \in (\mathbb{R}^d)^\mathcal{V}$  and  $\mathbf{P}$  is an orthogonal projection matrix onto the subspace  $\mathcal{L}^\perp$ . Matrix  $\mathbf{P}$  is given as follows:

$$\mathbf{P} = (\mathbf{I}_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^\top) \otimes \mathbf{I}_d, \quad (9)$$

where  $\mathbf{I}_p$  denotes  $p \times p$  identity matrix,  $\mathbf{1}_n = (1, \dots, 1) \in \mathbb{R}^n$ ,  $\otimes$  is a Kronecker product. Note that

$$\mathbf{P}^2 = \mathbf{P}. \quad (10)$$

With this notation, PNGD is presented as Algorithm 1.

A key property of Algorithm 1 is that it converges with the accelerated rate  $\mathcal{O}\left(\sqrt{L/\mu} \log\frac{1}{\epsilon}\right)$ . However, PNGD in each iteration calculates the matrix-vector multiplication  $\mathbf{P}\nabla F^*(z_g^k)$ , which requires *full averaging*, i.e., consensus, over all nodes of the communication network. In particular, this *can not be done in decentralized fashion*. In Section 5 we describe our algorithm ADOM, which in a certain sense mimics the behavior of Algorithm 1, but *can be implemented in a decentralized fashion*.

---

**Algorithm 1** PNGD: Projected Nesterov Gradient Descent

---

- 1: **input:**  $z^0 \in \mathcal{L}^\perp$ ,  $\alpha, \eta, \theta > 0$ ,  $\tau \in (0, 1)$
  - 2: set  $z_f^0 = z^0$
  - 3: **for**  $k = 0, 1, 2 \dots$  **do**
  - 4:    $z_g^k = \tau z^k + (1 - \tau) z_f^k$
  - 5:    $z^{k+1} = z^k + \eta \alpha (z_g^k - z^k) - \eta \mathbf{P} \nabla F^*(z_g^k)$
  - 6:    $z_f^{k+1} = z_g^k - \theta \mathbf{P} \nabla F^*(z_g^k)$
  - 7: **end for**
- 

## 4. Decentralized Communication

We now introduce the necessary notation, definitions and formalism to be able to describe our method. Compute nodes  $\mathcal{V} = [n]$  are connected through a communication network represented as a graph  $\mathcal{G}^k = (\mathcal{V}, \mathcal{E}^k)$ , where  $k \in \{0, 1, 2, \dots\}$  encodes time, and  $\mathcal{E}^k \subseteq \{(i, j) \in \mathcal{V} \times \mathcal{V} : i \neq j\}$  is the set of edges at time  $k$ . In this work we assume that the graph  $\mathcal{G}^k$  is undirected, that is,  $(i, j) \in \mathcal{E}^k$  implies  $(j, i) \in \mathcal{E}^k$ . We also assume that  $\mathcal{G}^k$  is connected. For each node  $i \in \mathcal{V}$  we consider a set of its neighbors at time step  $k$ :  $\mathcal{N}_i^k = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}^k\}$ . At time step  $k$ , each node  $i \in \mathcal{V}$  can communicate with the nodes from set  $\mathcal{N}_i^k$  only. This type of communication is known as *decentralized communication* in the literature.

### 4.1. Gossip Matrices

Decentralized communication between nodes is typically represented via a matrix-vector multiplication with a *gossip matrix*. For time-invariant networks such representations can be found in, e.g., (Kovalev et al., 2020b). For each time step  $k \in \{0, 1, 2, \dots\}$  consider a matrix  $\hat{\mathbf{W}}(k) \in \mathbb{R}^{n \times n}$  with the following properties:

1.  $\hat{\mathbf{W}}(k)$  is symmetric and positive semi-definite,
2.  $\hat{\mathbf{W}}(k)_{i,j} \neq 0$  if and only if  $(i, j) \in \mathcal{E}^k$  or  $i = j$ ,
3.  $\ker \hat{\mathbf{W}}(k) = \{(x_1, \dots, x_n) \in \mathbb{R}^n : x_1 = \dots = x_n\}$ .

Matrix  $\hat{\mathbf{W}}(k)$  is often called a *gossip matrix*. A typical example is the Laplacian of the graph  $\mathcal{G}^k$ . Consider also a linear map  $\mathbf{W}(k): (\mathbb{R}^d)^\mathcal{V} \rightarrow (\mathbb{R}^d)^\mathcal{V}$ , i.e.,  $nd \times nd$  matrix defined by  $\mathbf{W}(k) = \hat{\mathbf{W}}(k) \otimes \mathbf{I}_d$ . This matrix can be represented as a block matrix  $(\mathbf{W}(k)_{i,j})_{(i,j) \in \mathcal{V}^2}$ , where each block  $\mathbf{W}(k)_{i,j} = \hat{\mathbf{W}}(k)_{i,j} \mathbf{I}_d$  is a  $d \times d$  matrix proportional to  $\mathbf{I}_d$ . Matrix  $\mathbf{W}(k)$  satisfies similar properties to  $\hat{\mathbf{W}}(k)$ :

1.  $\mathbf{W}(k)$  is symmetric and positive semi-definite,
2.  $\mathbf{W}(k)_{i,j} \neq 0$  if and only if  $(i, j) \in \mathcal{E}^k$  or  $i = j$ ,
3.  $\ker \mathbf{W}(k) = \mathcal{L}$  or equivalently  $\text{range} \mathbf{W}(k) = \mathcal{L}^\perp$ .

With a slight abuse of language, in the rest of this paper we will refer to  $\mathbf{W}(k)$  as a *gossip matrix* as well.

### 4.2. Decentralized Communication as Multiplication with the Gossip Matrix

Decentralized communication of vectors  $x_1, \dots, x_n \in \mathbb{R}^d$  stored on the nodes among neighboring nodes at time step  $k$  can be represented as a multiplication of the  $nd$ -dimensional vector by matrix  $\mathbf{W}(k)$ . Indeed, consider  $x = (x_1, \dots, x_n) \in (\mathbb{R}^d)^\mathcal{V}$ ,  $y = (y_1, \dots, y_n) \in (\mathbb{R}^d)^\mathcal{V}$ , where each  $x_i$  is stored by node  $i \in \mathcal{V}$ , and let  $y = \mathbf{W}(k)x$ . One can observe that

$$y_i = \sum_{j=1}^n \hat{\mathbf{W}}(k)_{i,j} x_j = \sum_{j \in \mathcal{N}_i} \hat{\mathbf{W}}(k)_{i,j} x_j.$$

Hence, for each node  $i$ , vector  $y_i$  is a linear combination of vectors  $x_j$ , stored at the neighboring nodes  $j \in \mathcal{N}_i$ . This means that matrix-vector multiplications by matrix  $\mathbf{W}(k)$  can be computed in a decentralized fashion.

### 4.3. Condition Number of Time-Varying Networks

A condition number of the matrix  $\hat{\mathbf{W}}(k)$  is given as  $\frac{\lambda_{\max}(\hat{\mathbf{W}}(k))}{\lambda_{\min}^+(\hat{\mathbf{W}}(k))}$ , where  $\lambda_{\max}$  refers to the largest and  $\lambda_{\min}^+$  to the smallest positive eigenvalue. This quantity is known to be a measure of the connectivity of graph  $\mathcal{G}^k$ , and appears in convergence rates of many decentralized algorithms. In this work we assume that this condition number is bounded for all  $k \in \{0, 1, 2, \dots\}$ . In particular, we assume that there exist constants  $0 < \lambda_{\min}^+ < \lambda_{\max}$  such that

$$\lambda_{\min}^+ \leq \lambda_{\min}^+(\hat{\mathbf{W}}(k)) \leq \lambda_{\max}(\hat{\mathbf{W}}(k)) \leq \lambda_{\max}. \quad (11)$$

So, we assume that the worst case spectral behavior of the gossip matrices is bounded, and these bounds will later appear in our convergence rate for ADOM.

Relation (11) can be equivalently written in the form of a linear matrix inequality involving the gossip matrix  $\mathbf{W}(k)$ :

$$\lambda_{\min}^+ \mathbf{P} \preceq \mathbf{W}(k) \preceq \lambda_{\max} \mathbf{P}, \quad (12)$$

where  $\mathbf{P}$  is orthogonal projector onto subspace  $\text{range} \mathbf{W}^k = \mathcal{L}^\perp$  given by (9). Note that

$$\mathbf{P} \mathbf{W}(k) = \mathbf{W}(k) \mathbf{P} = \mathbf{W}(k). \quad (13)$$

By  $\chi$  we denote a bound on the condition number of matrices  $\mathbf{W}(k)$ ,  $k = 0, 1, 2, \dots$ , given by

$$\chi := \lambda_{\max} / \lambda_{\min}^+. \quad (14)$$

### 4.4. Decentralized Communication as a Compression Operator

We have just shown that decentralized communication at time step  $k$  can be represented as multiplication by the gossip matrix  $\mathbf{W}(k)$ . We will now show, and this is a key



insight which was the starting point of our work, that *decentralized communication can also be seen as the application of a contractive compression operator*.

Let  $\mathcal{Q}$  be a linear space. A mapping  $\mathcal{C}: \mathcal{Q} \rightarrow \mathcal{Q}$  is called a *compression operator* if there exists  $\delta \in (0, 1]$  such that

$$\|\mathcal{C}(z) - z\|^2 \leq (1 - \delta)\|z\|^2 \text{ for all } z \in \mathcal{Q}. \quad (15)$$

The following lemma shows that matrix-vector multiplication by gossip matrix  $\mathbf{W}(k)$  is a contractive compression operator acting on the subspace  $\mathcal{L}^\perp$ .

**Lemma 1.** *Let  $\sigma \in (0, 1/\lambda_{\max})$ ,  $k \in \{0, 1, 2, \dots\}$ . Then the following inequality holds for all  $z \in \mathcal{L}^\perp$ :*

$$\|\sigma \mathbf{W}(k)z - z\|^2 \leq (1 - \sigma \lambda_{\min}^+) \|z\|^2.$$

There is a natural question regarding Algorithm 1: can we replace the gradient  $\mathbf{P}\nabla F^*(z_g^k)$  on lines 5 and 6 with its compressed version  $\mathbf{W}(k)\mathbf{P}\nabla F^*(z_g^k)$ , or some modification thereof, and still obtain a good convergence result? Note that (13) implies  $\mathbf{W}(k)\mathbf{P}\nabla F^*(z_g^k) = \mathbf{W}(k)\nabla F^*(z_g^k)$ . In Section 5 we will provide a positive answer to this question.

Convergence of gradient-type methods with contractive compression operators satisfying (15) was studied in several recent papers. In particular, Stich & Karimireddy (2019); Karimireddy et al. (2019); Beznosikov et al. (2020); Gorbunov et al. (2020a) study a mechanism called *error feedback*, which allows to design variations with better convergence properties. However, these works do not study accelerated algorithms, nor provide any connections between compression and decentralized communication, which we do.

The only exception we are aware of is (Qian et al., 2020), which is the first work proposing an accelerated error compensated method. However, their ECLK method is more complicated than ours, uses the Katyusha momentum (Allen-Zhu, 2017; Kovalev et al., 2020a) instead of the Nesterov momentum we employ, and does not apply to decentralized optimization. Moreover, while for us it is crucial that the contractive property is enforced on a subspace only, Qian et al. (2020) require this property to hold globally.

## 5. ADOM: Algorithm and its Analysis

Armed with the notions and ideas described in preceding sections, we are now ready to present our method ADOM (Algorithm 2). As alluded to in the introduction, ADOM is a generalization of Algorithm 1 that can be implemented in a decentralized fashion. Indeed, our algorithm does *not* make use of matrix-vector multiplication by  $\mathbf{P}$  in the way Algorithm 1 does, which requires full averaging over the network. Instead, ADOM uses matrix-vector multiplication by the gossip matrix  $\mathbf{W}(k)$ , which represents a single decentralized communication round, as we discussed in Section 4.

---

### Algorithm 2 ADOM: Accelerated Decentralized Optimization Method

---

```

1: input:  $z^0 \in \mathcal{L}^\perp, m^0 \in (\mathbb{R}^d)^\nu, \alpha, \eta, \theta, \sigma > 0, \tau \in (0, 1)$ 
2: set  $z_f^0 = z^0$ 
3: for  $k = 0, 1, 2, \dots$  do
4:    $z_g^k = \tau z^k + (1 - \tau)z_f^k$ 
5:    $\Delta^k = \sigma \mathbf{W}(k)(m^k - \eta \nabla F^*(z_g^k))$ 
6:    $m^{k+1} = m^k - \eta \nabla F^*(z_g^k) - \Delta^k$ 
7:    $z^{k+1} = z^k + \eta \alpha (z_g^k - z^k) + \Delta^k$ 
8:    $z_f^{k+1} = z^k - \theta \mathbf{W}(k) \nabla F^*(z_g^k)$ 
9: end for

```

---

### 5.1. Design and Analysis of the New Algorithm

First, we mention two lemmas, which play an important role in the convergence analysis of Algorithm 2.

**Lemma 2.** *For  $\theta \leq \frac{\mu}{\lambda_{\max}}$  we have the inequality*

$$F^*(z_f^{k+1}) \leq F^*(z_g^k) - \frac{\theta \lambda_{\min}^+}{2} \|\nabla F^*(z_g^k)\|_{\mathbf{P}}^2. \quad (16)$$

**Lemma 3.** *For  $\sigma \leq \frac{1}{\lambda_{\max}}$  we have the inequality*

$$\begin{aligned} \|m^k\|_{\mathbf{P}}^2 &\leq \left(1 - \frac{\sigma \lambda_{\min}^+}{4}\right) \frac{4}{\sigma \lambda_{\min}^+} \|m^k\|_{\mathbf{P}}^2 \\ &\quad - \frac{4}{\sigma \lambda_{\min}^+} \|m^{k+1}\|_{\mathbf{P}}^2 + \frac{8\eta^2}{(\sigma \lambda_{\min}^+)^2} \|\nabla F^*(z_g^k)\|_{\mathbf{P}}^2. \end{aligned} \quad (17)$$

We now make a few remarks about the main steps of Algorithm 2 compared to Algorithm 1 and comment on some aspects of our convergence analysis and the role of the above lemmas in it:

- Line 4 of Algorithm 2 is unchanged compared to line 4 of Algorithm 1.
- Line 8 of Algorithm 2 corresponds to line 6 of Algorithm 1. Note that the analysis of Algorithm 1 requires an inequality of the type

$$F^*(z_f^{k+1}) \leq F^*(z_g^k) - \text{const} \cdot \|\nabla F^*(z_g^k)\|_{\mathbf{P}}^2.$$

Lemma 2 establishes a similar inequality for Algorithm 2.

- Together, lines 5, 6 and 7 of Algorithm 2 form an error feedback update, which we discussed in Section 4.4 when we interpreted a decentralized communication round as the application of a contractive compression operator. A key to the theoretical analysis of this update is to make use of the so-called *ghost iterate*  $\hat{z}^k = z^k + \mathbf{P}m^k$ . One can observe that the ghost iterate is updated as

$$\hat{z}^{k+1} = \hat{z}^k + \eta \alpha (z_g^k - z^k) - \eta \mathbf{P} \nabla F^*(z_g^k),$$

which is similar to the update on line 5 of Algorithm 1.

- Another key step in the analysis of Algorithm 2 is to bound the distance between the actual iterate  $z^k$  and the ghost iterate  $\hat{z}^k$ , which is equal to  $\|m^k\|_{\mathbb{P}}^2$ . This is done in Lemma 3 based on line 6 of Algorithm 2.

## 5.2. Main Convergence Theorem

Now, we are ready to present our main theorem.

**Theorem 1.** *Set parameters  $\alpha, \eta, \theta, \sigma, \tau$  of Algorithm 2 to*

$$\alpha = \frac{1}{2L}, \eta = \frac{2\lambda_{\min}^+ \sqrt{\mu L}}{7\lambda_{\max}}, \theta = \frac{\mu}{\lambda_{\max}}, \sigma = \frac{1}{\lambda_{\max}}, \text{ and}$$

$$\tau = \frac{\lambda_{\min}^+}{7\lambda_{\max}} \sqrt{\frac{\mu}{L}}. \text{ Then there exists } C > 0, \text{ such that}$$

$$\|\nabla F^*(z_g^k) - x^*\|^2 \leq C \left(1 - \frac{\lambda_{\min}^+}{7\lambda_{\max}} \sqrt{\frac{\mu}{L}}\right)^k. \quad (18)$$

Note that the rate is  $\mathcal{O}(\chi \kappa^{1/2} \log \frac{1}{\epsilon})$ , as previously advertised. Proofs of all our results are available in the appendix.

## 5.3. Comparison with Existing Algorithms

In this paper we compare our Algorithm 2 with current state-of-the-art algorithms for decentralized optimization over time-varying networks. While the Accelerated Penalty Method (APM) (Li et al., 2018) and Mudag (Ye et al., 2020) were originally designed for time-invariant networks, they can be easily extended to the time-varying case. Also note that DIGing (Nedic et al., 2017), Push-Pull Gradient Method (Pu et al., 2020) and PANDA (Maros & Jaldén, 2018) converge under slightly more general assumptions than those we used to analyze our method. However, these methods converge at a substantially slower rate due to the fact that they do not employ any acceleration mechanism. Moreover, to the best of our knowledge, no results improving the convergence rates of these algorithms under our assumptions exist in the literature.

## 6. Numerical Experiments

In this section we perform experiments with logistic regression for binary classification with  $\ell^2$  regularization. That is, our loss function has the form

$$f_i(x) = \frac{1}{m} \sum_{j=1}^m \log(1 + \exp(-b_{ij} a_{ij}^\top x)) + \frac{r}{2} \|x\|^2, \quad (19)$$

where  $a_{ij} \in \mathbb{R}^d$  and  $b_{ij} \in \{-1, +1\}$  are data points and labels,  $r > 0$  is a regularization parameter, and  $m$  is the number of data points stored on each node. In our experiments we use function `sklearn.datasets.make_classification` from scikit-learn library for dataset generation. We generate a number of datasets consisting of 10,000 samples, distributed to the  $n = 100$  nodes of the network with  $m = 100$  samples on each node. We vary  $r$  to obtain different values

of the condition number  $\kappa$ . We also vary the number of features  $d$ .

In order to simulate a time-varying network, we use geometric random graphs. That is, we generate  $n = 100$  nodes from the uniform distribution over  $[0, 1]^2 \subset \mathbb{R}^2$  and connect each pair of nodes whose distance is less than a certain *radius*. Since a geometric graph is likely to be disconnected when the radius is small, we enforce connectivity by adding a minimal number of edges. We obtain a sequence of networks  $\{\mathcal{G}^k\}_{k=0}^\infty$  by generating 1,000 random geometric graphs and switching between them in a cyclic way. For each  $k$ , matrix  $\mathbf{W}(k)$  is chosen to be the Laplacian of graph  $\mathcal{G}^k$  divided by its largest eigenvalue. We obtain different values of the time-varying network structure parameter  $\chi$  by choosing different values of the radius.

One potential problem with ADOM is that it has to calculate the dual gradient  $\nabla F^*(z_g^k)$ , which is known to be the solution of the following problem:

$$\nabla F^*(z_g^k) = \arg \min_{x \in (\mathbb{R}^d)^\nu} F(x) - \langle x, z_g^k \rangle. \quad (20)$$

In practice,  $\nabla F^*(z_g^k)$  may be hard to compute. In our experiments we solve this issue by calculating  $\nabla F^*(z_g^k)$  inexactly using  $T$  iterations of Gradient Descent (GD) or Accelerated Gradient Descent (AGD) initialized with the previous estimate of  $\nabla F^*(z_g^{k-1})$ . It turns out that it is sufficient to use  $T \leq 3$  to obtain a good convergence rate in practice.

### 6.1. Experiment 1: While DNM may diverge, ADOM is stable

We first compare ADOM with the Distributed Nesterov Method (DNM) of Rogozin et al. (2019). The condition number  $\kappa$  is set to 30, the number of features is  $d = 40$ . To calculate the dual gradient  $\nabla F^*(z)$  we use  $T = 3$  steps of AGD in ADOM and  $T = 30$  steps of AGD in DNM.

We switch between 2 networks every  $t$  iterations, where  $t \in \{50, 20, 10, 5\}$ . We use the following choice of networks: (i) two random geometric graphs with  $\chi \approx 400$ ; see Figure 2 (top row); (ii) two networks with ring and star topology with  $\chi \approx 1,000$ ; see Figure 2 (bottom row).

DNM diverges in 7 out of 8 cases presented in Figure 2, while ADOM converges in all cases. However, when DNM converges, it can converge faster than ADOM, since its communication complexity has better dependence on  $\chi$  ( $\sqrt{\chi}$  of DNM vs  $\chi$  of ADOM).

### 6.2. Experiment 2: Comparing ADOM with the state of the art: Mudag, Acc-DNGD and APM

We compare ADOM with the following algorithms for decentralized optimization over time-varying networks, all equipped with Nesterov acceleration: Mudag (Ye et al.,

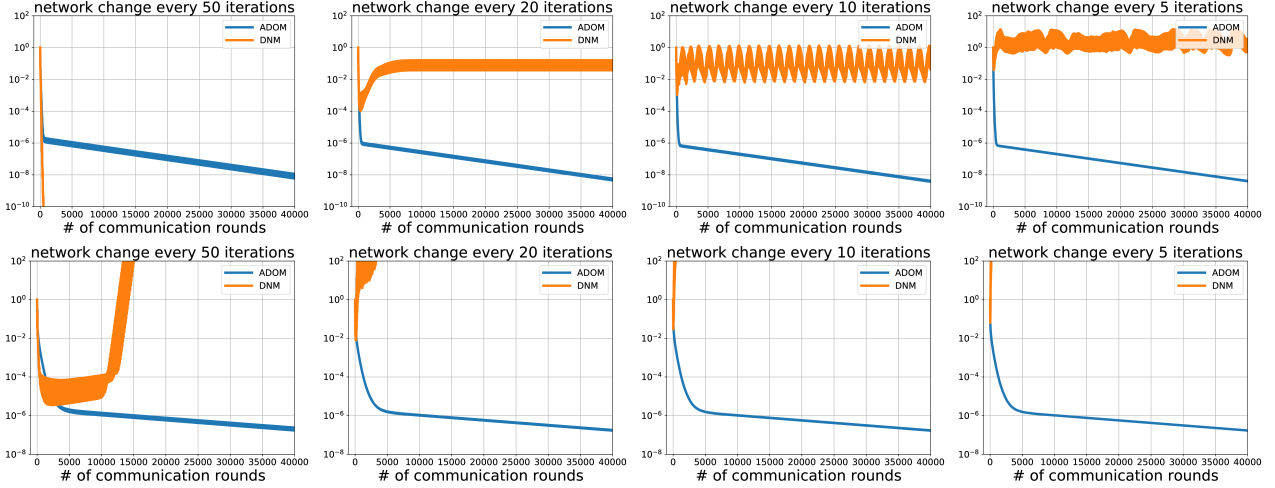


Figure 2. Comparison of DNM and ADOM on a problem with  $\kappa = 30$  and  $d = 40$ . **Top row:** We alternate between two geometric graphs ( $\chi \approx 400$ ). **Bottom row:** We alternate between two networks, one with a ring and the other with a star topology ( $\chi \approx 1000$ ).

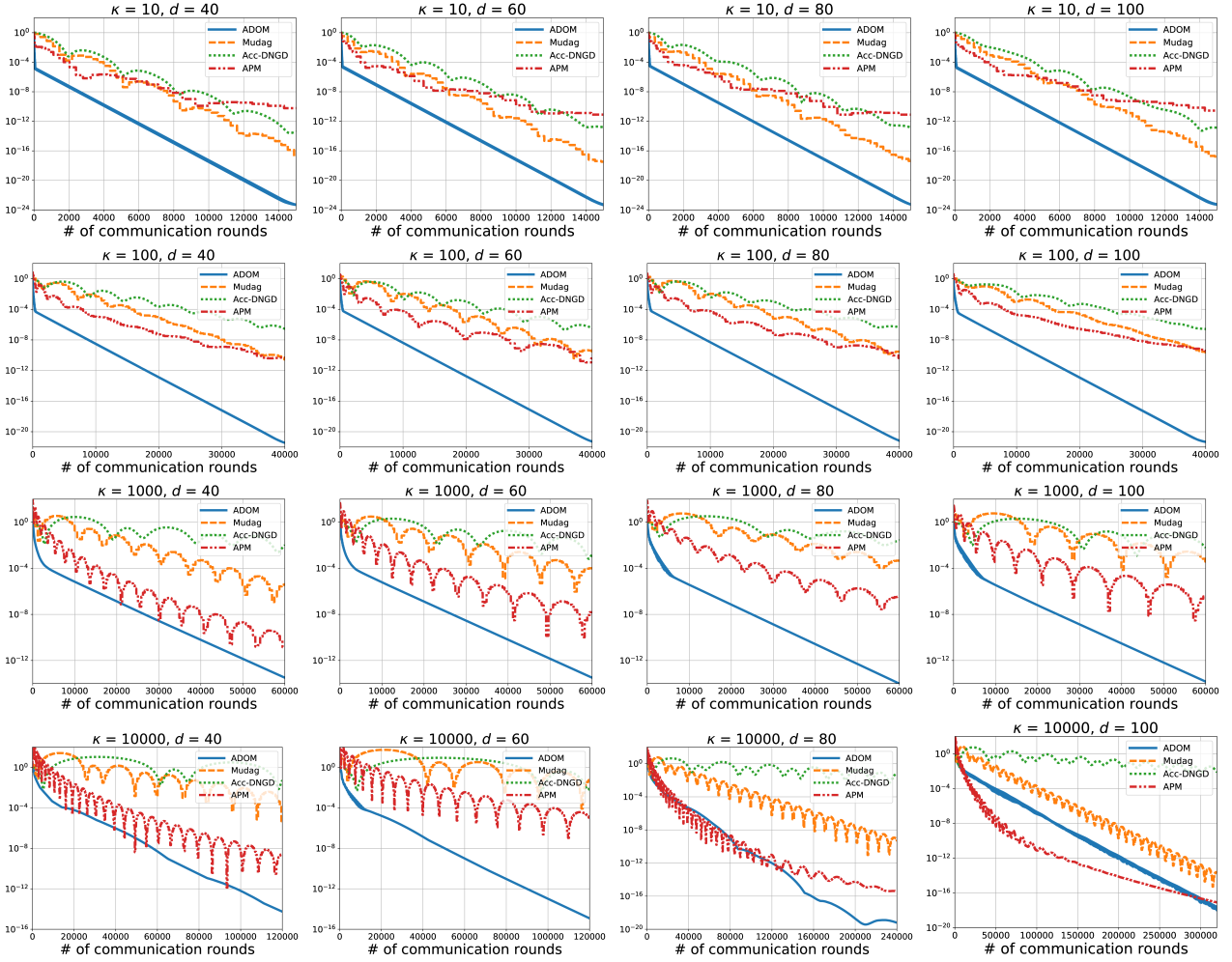


Figure 3. Comparison of Mudag, Acc-DNGD, APM and ADOM on problems with  $\chi \approx 30$ ,  $d \in \{40, 60, 80, 100\}$  and  $\kappa \in \{10, 10^2, 10^3, 10^4\}$ .

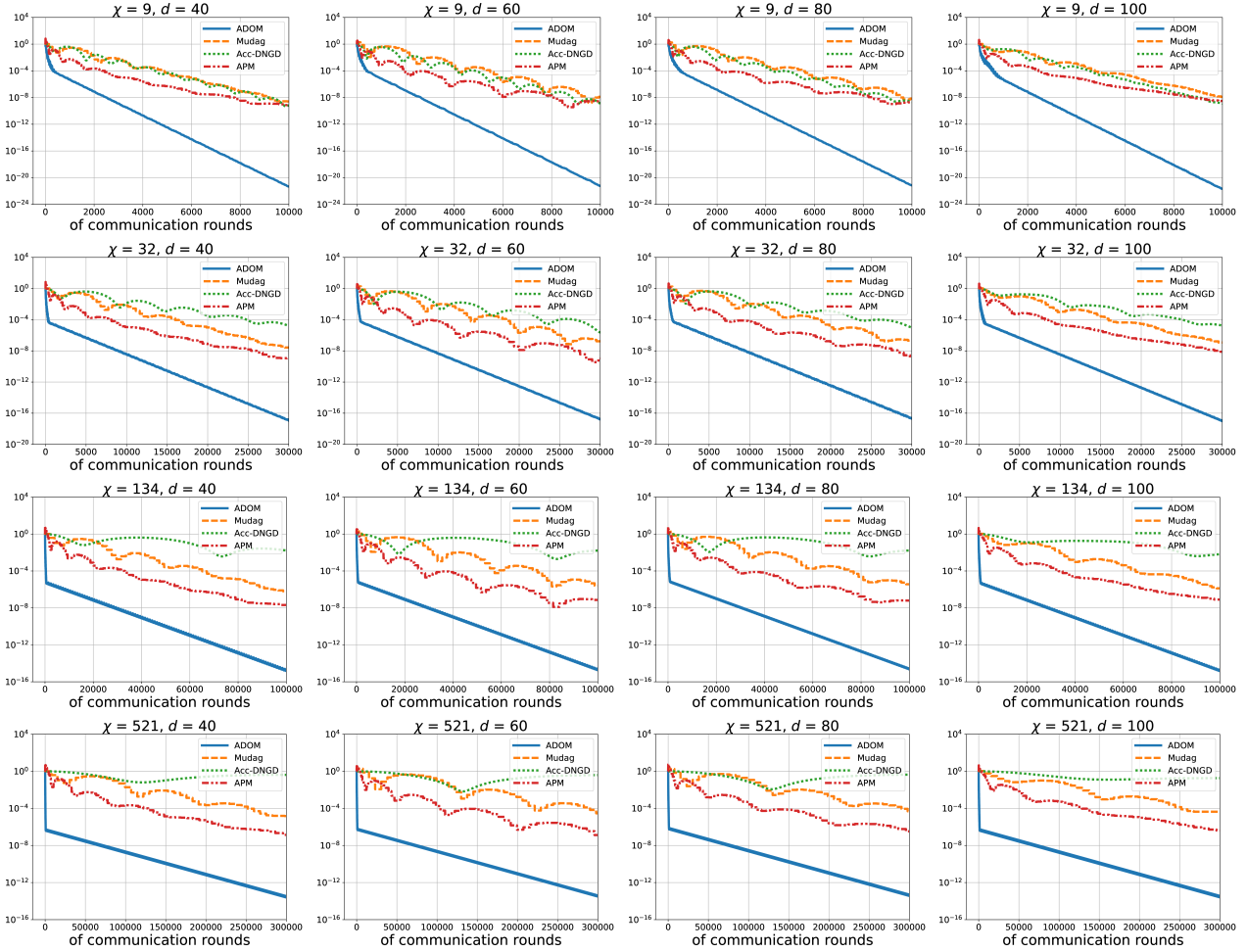


Figure 4. Comparison of Mudag, Acc-DNGD, APM and ADOM on problems with  $\chi \in \{9, 32, 134, 521\}$ ,  $d \in \{40, 60, 80, 100\}$  and  $\kappa = 100$ .

2020), Acc-DNGD (Qu & Li, 2019) and Accelerated Penalty Method (APM) (Li et al., 2018; Rogozin et al., 2020). We do not compare ADOM with PANDA (Maros & Jaldén, 2018) and DIGing (Nedic et al., 2017) because they are not accelerated and have very slow convergence rate both in theory and practice. We use  $T = 1$  iterations of GD to calculate  $\nabla F^*(z_g^k)$  in ADOM.

We generate random datasets with the number of features  $d \in \{40, 60, 80, 100\}$ . In Figure 3 we fix the network structure parameter  $\chi \approx 30$  and perform comparison for condition number  $\kappa \in \{10, 10^2, 10^3, 10^4\}$ . In Figure 4 we fix  $\kappa = 100$  and perform comparison for  $\chi \in \{9, 32, 134, 521\}$ .

Overall, ADOM is better than the contenders. Acc-DNGD performs worse as the values of  $\chi$  and  $\kappa$  grow since it has the worst dependence on them. One can also observe that APM suffers from sub-linear convergence, which becomes clear as the number of iterations grows (see bottom row of Fig-

ure 3) since its communication complexity is proportional to  $\log^2 \frac{1}{\epsilon}$ .

### Acknowledgments

The work of D. Kovalev, E. Shulgin and P. Richtárik was supported by the KAUST Baseline Research Funding Scheme. The work of A. Rogozin and A. Gasnikov was supported by the Russian Science Foundation (project 21-71-30005).

### References

Allen-Zhu, Z. Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1200–1205. ACM, 2017.

Bazerque, J. A. and Giannakis, G. B. Distributed spectrum sensing for cognitive radio networks by exploiting



- sparsity. *IEEE Transactions on Signal Processing*, 58(3): 1847–1862, 2009.
- Beck, A., Nedić, A., Ozdaglar, A., and Teboulle, M. An  $o(1/k)$  gradient method for network resource allocation problems. *IEEE Transactions on Control of Network Systems*, 1(1):64–73, 2014.
- Beznosikov, A., Horváth, S., Richtárik, P., and Safaryan, M. On biased compression for distributed learning. *arXiv:2002.12410*, 2020.
- Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Gan, L., Topcu, U., and Low, S. H. Optimal decentralized protocol for electric vehicle charging. *IEEE Transactions on Power Systems*, 28(2):940–951, 2012.
- Giselsson, P., Doan, M. D., Keviczky, T., De Schutter, B., and Rantzer, A. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3):829–833, 2013.
- Gorbunov, E., Kovalev, D., Makarenko, D., and Richtárik, P. Linearly converging error compensated SGD. *Advances in Neural Information Processing Systems*, 33, 2020a.
- Gorbunov, E., Rogozin, A., Beznosikov, A., Dvinskikh, D., and Gasnikov, A. Recent theoretical advances in decentralized distributed convex optimization. *arXiv preprint arXiv:2011.13259*, 2020b.
- Karimireddy, S. P., Rebjock, Q., Stich, S. U., and Jaggi, M. Error feedback fixes SignSGD and other gradient compression schemes. *arXiv preprint arXiv:1901.09847*, 2019.
- Kolar, M., Song, L., Ahmed, A., and Xing, E. P. Estimating time-varying networks. *The Annals of Applied Statistics*, 4(1):94–123, 2010.
- Konečný, J., McMahan, H. B., Yu, F., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: strategies for improving communication efficiency. In *NIPS Private Multi-Party Machine Learning Workshop*, 2016.
- Kovalev, D., Horváth, S., and Richtárik, P. Don't jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop. In *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, 2020a.
- Kovalev, D., Salim, A., and Richtárik, P. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. *Advances in Neural Information Processing Systems*, 33, 2020b.
- Kovalev, D., Koloskova, A., Jaggi, M., Richtárik, P., and Stich, S. A linearly convergent algorithm for decentralized optimization: Sending less bits for free! In *The 24th International Conference on Artificial Intelligence and Statistics (AISTATS 2021)*, 2021.
- Li, H., Fang, C., Yin, W., and Lin, Z. A sharp convergence rate analysis for distributed accelerated gradient methods. *arXiv preprint arXiv:1810.01053*, 2018.
- Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020a.
- Li, Z., Kovalev, D., Qian, X., and Richtárik, P. Acceleration for compressed gradient descent in distributed and federated optimization. In *International Conference on Machine Learning*, 2020b.
- Maros, M. and Jaldén, J. Panda: A dual linearly converging method for distributed optimization over time-varying undirected graphs. In *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6520–6525. IEEE, 2018.
- McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.
- Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL [www.graphlearning.io](http://www.graphlearning.io).
- Nedic, A., Olshevsky, A., and Shi, W. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4): 2597–2633, 2017.
- Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.
- Pu, S., Shi, W., Xu, J., and Nedic, A. Push-pull gradient methods for distributed optimization in networks. *IEEE Transactions on Automatic Control*, 2020.
- Qian, X., Richtárik, P., and Zhang, T. Error compensated distributed SGD can be accelerated. *arXiv preprint arXiv:2010.00091*, 2020.
- Qu, G. and Li, N. Accelerated distributed nesterov gradient descent. *IEEE Transactions on Automatic Control*, 2019.

- Rabbat, M. and Nowak, R. Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pp. 20–27, 2004.
- Rockafellar, R. T. *Convex analysis*, volume 36. Princeton university press, 1970.
- Rogozin, A., Uribe, C., Gasnikov, A., Malkovskii, N., and Nedich, A. Optimal distributed convex optimization on slowly time-varying graphs. *IEEE Transactions on Control of Network Systems*, 2019.
- Rogozin, A., Lukoshkin, V., Gasnikov, A., Kovalev, D., and Shulgin, E. Towards accelerated rates for distributed optimization over time-varying networks. *arXiv preprint arXiv:2009.11069*, 2020.
- Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal algorithms for smooth and strongly convex distributed optimization in networks. *arXiv preprint arXiv:1702.08704*, 2017.
- Stich, S. U. and Karimireddy, S. P. The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*, 2019.
- Ye, H., Luo, L., Zhou, Z., and Zhang, T. Multi-consensus decentralized accelerated gradient descent. *arXiv preprint arXiv:2005.00797*, 2020.
- Zadeh, L. A. Time-varying networks, i. *Proceedings of the IRE*, 49(10):1488–1503, 1961.