2024 IEEE 25th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC) | 979-8-3503-9318-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/SPAWC60668.2024.10694261

Fully-distributed optimization with Network Exact Consensus-GIANT

Alessio Maritan*, Ganesh Sharma†, Subhrakanti Dey‡, and Luca Schenato*
*Department of Information Engineering, University of Padova, Italy
†Hamilton Institute, Maynooth University, Ireland
‡Department of Electrical Engineering, Uppsala University, Sweden
email: alessio.maritan@phd.unipd.it, ganesh.sharma.2019@mumail.ie, subhrakanti.dey@angstrom.uu.se, schenato@dei.unipd.it

***Abstract*—We consider a fully-distributed optimization problem involving multiple collaborative agents, where the global objective is to minimize a sum of local cost functions. Agents are part of a communication network and can only exchange information with their neighbors. We introduce a novel optimization algorithm called NEC-GIANT, which improves over both GIANT, a popular federated learning algorithm, and Network-GIANT, our previously proposed fully-distributed counterpart of GIANT. NEC-GIANT extends GIANT to the fully-distributed scenario, removing the need for a central server to orchestrate the agents. Unlike the existing Network-GIANT, which suffers from the inefficiency of standard asymptotic consensus, the novel NEC-GIANT is based on finite-time distributed consensus and retains all the convergence properties of the original GIANT. Numerical simulations prove the efficiency and superiority of the proposed algorithm in terms of both iterations and machine run-time.**

***Index Terms*—distributed optimization, gradient tracking, finite-time consensus, network learning, Newton-type algorithms**

I. Introduction

Distributed optimization is attracting growing interest in various fields, motivated by the limitations of centralized optimization, the prevalence of data islands, the increasing privacy concerns, and the advances in communication technologies. Various sectors can benefit from distributed optimization, such as the Internet of Things, autonomous decision making and smart cities. In particular, considerable efforts have been invested in the development of a specific distributed framework called federated learning. In the latter, a central server orchestrates interactions among nodes to facilitate iterative algorithms, usually by aggregating and broadcasting data. Both gradient-based federated optimization [1] and Hessian-based federated optimization have been researched, with several references proposing approximate Newton-type methods e.g. DANE [2], GIANT [3], DANDA [4], FedNL [5], SHED [6].

Scenarios where a central server is not available require resorting to network (also called fully-distributed) optimization algorithms, where nodes communicate with each other in a peer-to-peer fashion to establish consensus. A wide range of communication protocols for achieving distributed consensus can be found in the literature. For example, gossip-based algorithms where nodes communicate with one or few randomly chosen neighbors have been investigated in [7], [8]. Distributed averaging based on linear iterations, with a suitable weighting matrix, has been implemented in both fixed [9] and time-varying network topologies [10]. Unlike traditional consensus algorithms that only provide asymptotic convergence to the average of the local variables, the finite-time consensus method in [11] allows one to compute the exact average in a finite number of steps. Similar results are achieved by the finite-time gossip protocols in [12], [13]. References [14], [15] investigate the finite-time consensus properties of some classes of network topology, in some cases by decomposing static graphs into sequences of graphs, and derive conditions under which consensus is reached in $\log_2(N)$ steps, where N is the number of nodes. Finite-time consensus has been investigated in gradient-based distributed optimization, for example, in [16].

Fully-distributed gradient-based optimization algorithms has been studied in a number of works, e.g. [17], [18], while network versions of ADMM have been explored in [19], [20]. To exploit the curvature of the objective function and achieve faster convergence, several second-order fully-distributed algorithms have been designed, such as NRC [21], Newton Tracking [22], Network-DANE [23], ESOM [24] and Network-GIANT [25]. Despite all such efforts, in general, the convergence of fully distributed algorithms is much slower than that of federated algorithms, which may rely on a central server to calculate averages in a single step. In contrast, fully distributed algorithms typically replace central aggregation with asymptotic consensus protocols, which limit the overall convergence rate to be at most linear.

Contributions: We introduce a novel fully-distributed optimization algorithm, that we call Network Exact Consensus-GIANT (NEC-GIANT). In particular, we improve upon our previously proposed Network-GIANT [25], which mimics the behavior of the federated learning algorithm GIANT [3] in the fully-distributed setting. The existing Network-GIANT suffers from a convergence slowdown compared to GIANT, and this is due to the use of standard asymptotic consensus. In this paper, we overcome this limitation by leveraging finite-

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 18/CRT/6049 and Swedish Research Council (VR) Grant 2023-04232.

A. Maritan is supported by the industrial scholarship PNRR DM352-2022 partially funded by Maschio Gaspardo S.p.A.

time distributed consensus, recovering the original iteration complexity of GIANT. More in general, we retain all the theoretical properties of GIANT, compensating the absence of the central server with additional short-range communications between neighbors. The proposed NEC-GIANT takes the best of both worlds: the appealing convergence guarantees of the federated GIANT, and the fully-distributed property of Network-GIANT. We test the practical performance of the proposed algorithm through numerical experiments using standard datasets, providing evidence of its efficiency.

Organization: The remainder of the paper is organized as follows. Section 2 provides a rigorous formulation of the problem under consideration. Section 3 introduces the preliminary concepts and theoretical tools underlying our algorithm, presented in Section 4. Section 5 contains numerical simulations on two standard datasets, followed by a final summary in Section 6.

Notation: For a generic variable x, we use x_i^k to denote the local copy possessed by agent i at iteration k. $\|\cdot\|$ is the Euclidean norm, I is the identity matrix and the superscript ^{T} indicates the transpose of the argument. The d-dimensional vector whose components are all equal to 1 is denoted by $\mathbb{1}$.

II. Problem Formulation

Consider a communication network that can be modeled by a connected and time-invariant graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, 2, ..., N\}$ denotes the set of nodes and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ the set of bidirectional edges connecting the nodes. Nodes can only directly communicate with their 1-hop neighbors. The network can be equivalently described by a weight matrix $P \in \mathbb{R}^{N\times N}$, where the element p_{ij} is positive if there is an edge $(i, j) \in \mathcal{E}$ and zero otherwise. The consensus matrix P is chosen symmetric and doubly stochastic, which implies that $P\mathbb{1} = \mathbb{1}$ and the eigenvalues of P lie in $(-1, 1]$. It is possible to construct this matrix in a distributed way, for example using the Metropolis algorithm [26].

In this setting, we consider a fully-distributed version of the unconstrained optimization problem in [3]:

$$f(x^\star) = \min_{x\in\mathbb{R}^d} \left\{ f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x) \right\}, \tag{1}$$

$$f_i(x) = \frac{1}{m}\sum_{j=1}^{m} l_{ij}(x^T a_{ij}) + \frac{\gamma}{2}\|x\|^2 . \tag{2}$$

Each node $i \in \mathcal{N}$ owns m data samples $\{a_{ij}\}$ $j \in \{1, ..., m\}$, each associated with a convex, twice differentiable and smooth loss function $l_{ij}(\cdot)$. The overall cost function at node i is given by the sum of the local empirical error and a regularization term, and is denoted by $f_i(x)$. The following assumption provides a rigorous characterization of the objective functions.

Assumption 1. *(a) The loss functions $l_{ij}(\cdot)$ are convex, twice differentiable and smooth $\forall i \in \mathcal{N}$, $\forall j \in \{1, ..., m\}$.*

(b) The global objective $f(\cdot)$ is strongly convex and has Lipschitz continuous Hessian, i.e. there exists a constant L such that $\forall x, x' \in \mathbb{R}^d$ $\left\|\nabla^2 f(x) - \nabla^2 f(x')\right\|_2 \leq L\|x - x'\|$.

The above assumption is customary within the realm of machine learning and convex optimization. In particular, Assumption 1(a) is met by numerous conventional cost functions, such as the least squares cost function used in linear regression and the logistic regression loss. Assumption 1(b) guarantees the uniqueness of the minimizer of (1) and is standard in second-order optimization.

III. Algorithm description

We first introduce some preliminary concepts, namely the working principles behind the algorithms GIANT [3] and Network-GIANT [25] and the finite-time distributed consensus method [11]. Next, we show how to combine these ingredients and design the novel NEC-GIANT algorithm.

A. GIANT and Network-GIANT

GIANT [3] is a popular optimization algorithm that solves the convex optimization problem (1) in the federated setting, where all nodes are connected in a star topology to a central server. GIANT updates the decision vector by descending along an approximate Newton direction, obtained by preconditioning the exact global gradient with the inverse of the harmonic mean of the local Hessian matrices:

$$x^{k+1} = x^k - \epsilon\frac{1}{N}\sum_{i=1}^{N}\left[\nabla^2 f_i(x^k)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(x^k)\right)\right]. \tag{3}$$

Neglecting the selection of the stepsize ϵ, each iteration involves the following steps: *(i)* the server broadcasts the current x^k, *(ii)* nodes transmit their local gradient $\nabla f_i(x^k)$, *(iii)* the server averages them and broadcasts the global gradient, *(iv)* nodes transmit their local approximate Newton direction (the term inside the square brackets), *(v)* the server aggregates these directions and updates the decision vector. GIANT enjoys a linear-quadratic convergence rate and was shown to outperform most federated algorithms for convex optimization.

Network-GIANT [25] extends GIANT to the peer-to-peer network setup, removing the need for a central server. To do so, steps *(ii)* and *(iii)* are replaced with average consensus tracking, introducing the auxiliary local variables w_i and performing the update below, which involves a consensus step:

$$w_i^{k+1} = \sum_{j=1}^{N} p_{ij}\left(w_j^k + \nabla f_j(x_j^k) - \nabla f_j(x_j^{k-1})\right). \tag{4}$$

If the differences $\nabla f_j(x_j^k) - \nabla f_j(x_j^{k-1})$ are sufficiently small compared to the consensus rate of the matrix P, then all w_i asymptotically converge to the global gradient. Steps from *(iv)* to *(i)* are instead substituted with the following fully-distributed update, which again involves a consensus step:

$$x_i^{k+1} = \sum_{j=1}^{N} p_{ij}\left(x_j^k - \epsilon\nabla^2 f_j(x_j^k)^{-1} w_j^{k+1}\right). \tag{5}$$

The consensus protocol used in Network-GIANT only provides asymptotic convergence to the average of the local variables, meaning that the exact average is never reached in

practice, as this would require an infinite number of steps. To alleviate this problem, a variant of Network-GIANT performs $K \geq 1$ consensus steps in (5). If K is not sufficiently large, the estimated average may be very far from the actual one, especially in the first iterations of the algorithm, introducing errors and slowing down the overall convergence.

B. Finite-Time Distributed Consensus

We now introduce the finite-time distributed consensus (FTDC) procedure in [11], which allows one to compute the exact average of local variables with a finite number of communication rounds between neighbors. At the end of the protocol each node knows the exact average, similarly to what happens in the federated setting but without the need of a central server.

The requirements that a consensus matrix P must satisfy to attain FTDC are the following: a) P has a simple eigenvalue at 1, and all other eigenvalues have magnitude strictly less than 1, b) The left and right eigenvectors of P corresponding to the eigenvalue 1 are $\frac{1}{N}\mathbb{1}^T$ and $\mathbb{1}$, respectively.

The theoretical concept underlying the protocol is the minimum polynomial of the matrix P, defined as the unique monic polynomial of smallest degree $D+1 \leq N$ that satisfies

$$P^{D+1} + \alpha_D P^D + \cdots + \alpha_1 P + \alpha_0 I = 0. \tag{6}$$

Assume that we want to calculate the average of a generic local variable s_i in a distributed manner. Let $s = [s_1 \ \cdots \ s_N]^T$ be the stack of the local variables, so that a single step of the FTDC protocol can be written as $s[t+1] = Ps[t]$. Since $s[D+1] = P^{D+1}s[0]$, taking P^{D+1} from (6) and considering a generic step $t \in \mathbb{N}$ we obtain the identity

$$s_i[t+D+1] + \alpha_D s_i[t+D] + \cdots + \alpha_1 s_i[t+1] + \alpha_0 s_i[t] = 0. \tag{7}$$

Equation (7) shows that the values taken by a local variable in the last $D+1$ consensus steps contain all the information needed to compute all its future values, without the need to actually perform further consensus steps. Taking the Z-transform of (7) and applying the final value theorem gives a closed-form expression of the average:

$$\frac{1}{N}\mathbb{1}^T s[0] = \lim_{t\to\infty} s_i[t] = \frac{\begin{bmatrix} s_i[D] & s_i[D-1] & \cdots & s_i[0] \end{bmatrix} G}{\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} G},$$

where

$$G = \begin{bmatrix} 1 \\ 1+\alpha_D \\ 1+\alpha_{D-1}+\alpha_D \\ \vdots \\ 1+\sum_{j=1}^{D}\alpha_j \end{bmatrix}. \tag{8}$$

C. NEC-GIANT

In this section, we propose a new algorithm called Network Exact Consensus-GIANT, which extends the federated algorithm GIANT [3] to the fully distributed scenario, keeping the convergence properties and iteration complexity unchanged.

We first point out the weakness of the existing fully-distributed counterpart of GIANT: Network-GIANT does not use exact averages of local variables, but rather estimates that may be very inaccurate, especially at the beginning of the algorithm. In fact, in Network-GIANT the global gradient is estimated using average consensus tracking, and the global approximate Newton direction is estimated indirectly by performing one or few steps of standard consensus on the updated decision variable. The consensus in (4) and (5) is only asymptotically convergent and therefore constitutes a performance bottleneck.

To overcome this problem, we propose an improved fully-distributed version of GIANT based on the FTDC protocol in [11]. The latter allows us to easily compute exact averages of local quantities in the absence of a central server, preserving the original properties of GIANT. In particular, we compute the exact global gradient by performing FTDC directly on local gradients. Unlike Network-GIANT, that relies on average consensus tracking, we do not require auxiliary variables that also entail additional computations and storage. Similarly, the global approximate Newton direction is computed exactly with a call to the FTDC subroutine, and is used to update the decision vector locally in parallel. This results in a simple and intuitive pseudocode, shown in Algorithm 2. Similarly to Network-GIANT, when descending along the approximate Newton direction, we allow for a stepsize $\epsilon > 0$. In comparison, [3] proposes and analyses GIANT with unit stepsize, and relies on backtracking line search for the numerical tests.

Algorithm 1 $\texttt{FTDC}(s_i[0],\, G,\, P)$ (seen from node i)

Input: initial value $s_i[0]$ of node i, coefficient vector $G \in \mathbb{R}^{D+1}$, consensus matrix $P \in \mathbb{R}^{N\times N}$.

for each step $t = 0, \ldots, D$ **do**
  for each node $i \in \mathcal{N}$ **in parallel do**
    $s_i[t+1] = \sum_{j=1}^{N} p_{ij} s_j[t]$
  end for
end for

$$\texttt{FTDC}(s_i[0],\, G,\, P) = \frac{\begin{bmatrix} s_i[D] & s_i[D-1] & \cdots & s_i[0] \end{bmatrix} G}{\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} G}$$

Algorithm 2 NEC-GIANT

Initialize:
  Arbitrary $x_i^0 = x_1^0 \in \mathbb{R}^d \ \forall i \in \mathcal{N}$,
  stepsize $\epsilon > 0$, consensus matrix P.

Compute the distinct eigenvalues $\{\lambda_0, ..., \lambda_D\}$ of P.
Compute the coefficients of the minimal polynomial $\prod_{i=0}^{D}(p-\lambda_i) = p^{D+1} + \alpha_D p^D + \cdots + \alpha_1 p + \alpha_0$.
Compute G as in (8).

for each iteration $k = 0, 1, \ldots$ **do**
  for each node $i \in \mathcal{N}$ **in parallel do**
    $g_i^k = \texttt{FTDC}\left(\nabla f_i(x_i^k),\, G,\, P\right)$
    $\eta_i^k = \texttt{FTDC}\left(\nabla^2 f_i(x_i^k)^{-1} g_i^k,\, G,\, P\right)$
    $x_i^{k+1} = x_i^k - \epsilon \eta_i^k$
  end for
end for

Algorithm 1 and the initialization in Algorithm 2 adapt the FTDC protocol in [11] to our specific use case. Being symmetric and doubly-stochastic, the weight matrix P automatically satisfies the FTDC requirements in Section III-B and eases the computation of the minimum polynomial. Each call to the FTDC subroutine involves $D+1$ communication rounds between 1-hop neighboring nodes, where $D+1$ is the number of distinct eigenvalues of P.

IV. Theoretical analysis

The key point in the associated convergence analysis is that each iteration of NEC-GIANT is totally equivalent to one of GIANT, and therefore the convergence trajectories of the two algorithms are identical. The only difference between the two algorithms is the way the averages are computed: while GIANT requires a central aggregator server, NEC-GIANT achieves the same result in a fully-distributed manner. This is formalized by the following proposition.

Proposition 1. *At the end of each iteration, the quantities computed by NEC-GIANT and GIANT (the global gradient, the global approximate Newton direction, and the updated decision vector) are identical, provided that both algorithms employ the same stepsize.*

Proof sketch. The proof follows from the following facts: *(i)* all nodes are initialized with the same arbitrary x_i^0 and perform the same updates in parallel, *(ii)* NEC-GIANT follows the same steps as GIANT, computing the same intermediate quantities and applying the same update rule, *(iii)* the FTDC protocol returns the exact average of the local quantities. □

As a consequence, in Algorithm 2 it holds $g_i^k = g_1^k$, $\eta_i^k = \eta_1^k$ and $x_i^k = x_1^k$ for all nodes $i \in \mathcal{N}$ and for all iterations $k \in \mathbb{N}$. NEC-GIANT also retains the linear-quadratic local convergence property of GIANT, in particular, for problem (1), with a communication overhead given as follows:

Proposition 2. *Each iteration of NEC-GIANT involves* $2(D-1)$ *additional transmissions per node compared to GIANT, where* $D+1 \leq N$ *is the number of distinct eigenvalues of the consensus matrix* P.

The above result follows from the following facts: *(i)* in GIANT, computing an average at the central server and broadcasting the result involves two transmissions per node, *(ii)* each call to the FTDC routine using the matrix P involves $D+1$ transmissions per agent, *(iii)* two averages per iteration are computed in both algorithms. We remark that the additional communications are typically short-range as they cross a single edge of the network, unlike in federated learning where all clients have to reach a possibly far central server. When the number of nodes N is large, $D+1$ may also be quite large. This potential disadvantage of the NEC-GIANT algorithm can be circumvented by designing an appropriate sequence of consensus matrices as done in [14], where only $\log(N)$ consensus steps are needed. This research direction is left for future work.

V. Numerical Analysis

This section reports the numerical experiments conducted in order to empirically evaluate the convergence performance of the proposed NEC-GIANT.

Our experiments focus on distributed binary classification via regularized logistic regression, so that $l_{ij} = \log(1 + e^{-b_{ij}(x^T a_{ij})})$ in (2), where a_{ij} is the input feature vector and b_{ij} is the corresponding target response. We use two widely recognized datasets: Covertype [27] with $d = 54$, and a compressed version of MNIST [28] where the feature size is brought down to $d = 300$ using Principal Component Analysis. We consider a network with $N = 20$ nodes and we build the mixing matrix P using the Metropolis weights [26].

The proposed NEC-GIANT is compared with GIANT [3] and Network-GIANT [25], and centralised Newton Method is shown for reference. Each call to the FTDC protocol employed by NEC-GIANT involves $D+1 = 20$ consensus steps. As for Network-GIANT, we denote with k_1 and k_2 the number of consensus steps performed in (4) and (5), respectively. For simplicity, we use a constant step size ϵ for all algorithms.

Figures 1 and 2 show the results of our tests. The training loss is $\left[\frac{1}{n}\sum_{i=1}^{n} f_i(x_i) - f(x^\star)\right] / f(x^\star)$, where $x^\star$ denotes the global minimum, and is plotted against the number of iterations and the machine runtime. As expected, the convergence trajectory of NEC-GIANT is identical to that of GIANT up to numerical precision in terms of number of iterations, and the additional communications due to FTDC only increase the machine runtime to a limited extent. The convergence rate of Network-GIANT is initially very similar to that of GIANT and NEC-GIANT, but shows a slowdown once a certain error is reached. The performance of NEC-GIANT on the test dataset, omitted due to space limitations, is similar to that of Network-GIANT, as reported in [25]. In summary, NEC-GIANT efficiently circumvents the absence of a central server and shows superior performance compared to the existing Network-GIANT.

VI. Conclusions

This paper introduces the novel optimization algorithm NEC-GIANT, that extends to the fully-distributed scenario the federated algorithm GIANT. Unlike the existing Network-GIANT, which is also a fully-distributed algorithm inspired by GIANT, NEC-GIANT preserves GIANT's convergence properties unchanged. The peculiarity of NEC-GIANT is the use of finite-time distributed consensus, which allows the dependence on the central server to be removed without any performance degradation, effectively bridging the gap between federated and fully distributed scenarios. Numerical tests confirm the superior convergence performance of the proposed algorithm.

References

[1] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[2] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *International Conference on Machine Learning*, 2014.

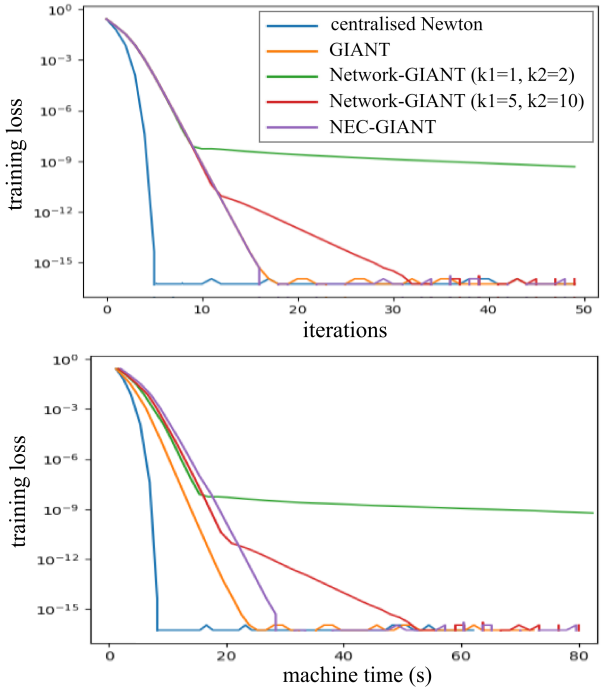

Fig. 1: Tests on the dataset Covertype [27].

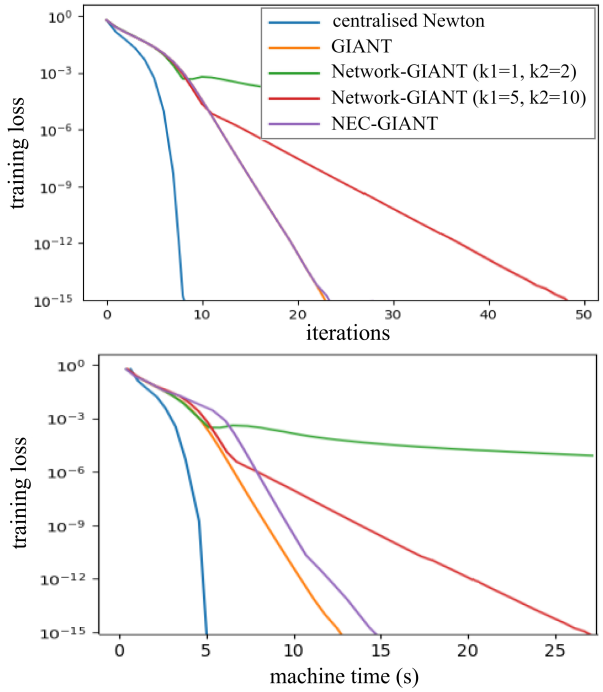

Fig. 2: Tests on the dataset MNIST [28].

[3] S. Wang, F. Roosta, P. Xu, and M. W. Mahoney, "Giant: Globally improved approximate newton method for distributed optimization," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[4] G. Sharma and S. Dey, "On analog distributed approximate Newton with determinantal averaging," in *2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2022, pp. 1–7.

[5] M. Safaryan, R. Islamov, X. Qian, and P. Richtárik, "FedNL: Making newton-type methods applicable to federated learning," *arXiv preprint arXiv:2106.02969*, 2021.

[6] N. Dal Fabbro, S. Dey, M. Rossi, and L. Schenato, "SHED: A newton-type algorithm for federated learning based on incremental hessian eigenvector sharing," *Automatica*, vol. 160, p. 111460, 2024.

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

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

[9] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[10] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.

[11] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *2007 American Control Conference*. IEEE, 2007, pp. 711–716.

[12] G. Shi, B. Li, M. Johansson, and K. H. Johansson, "Finite-time convergent gossiping," *IEEE/ACM Transactions on Networking*, 2016.

[13] J. Chen, S. Wang, L. Carin, and C. Tao, "Finite-time consensus learning for decentralized optimization with nonlinear gossiping," no. arXiv:2111.02949, Nov. 2021, arXiv:2111.02949 [cs].

[14] E. D. H. Nguyen, X. Jiang, B. Ying, and C. A. Uribe, "On graphs with finite-time consensus and their use in gradient tracking," no. arXiv:2311.01317, Nov. 2023, arXiv:2311.01317 [math]. [Online]. Available: http://arxiv.org/abs/2311.01317

[15] B. Ying, K. Yuan, Y. Chen, H. Hu, P. PAN, and W. Yin, "Exponential graph is provably efficient for decentralized deep training," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, p. 13975–13987.

[16] T. Charalambous, Y. Yuan, T. Yang, W. Pan, C. N. Hadjicostis, and M. Johansson, "Distributed finite-time average consensus in digraphs in the presence of time delays," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 4, pp. 370–381, 2015.

[17] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[18] D. Jakovetić, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, 2014.

[19] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, 2014.

[20] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2014.

[21] D. Varagnolo, F. Zanella, A. Cenedese, G. Pillonetto, and L. Schenato, "Newton-Raphson consensus for distributed convex optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 994–1009, 2015.

[22] J. Zhang, Q. Ling, and A. M.-C. So, "A Newton tracking algorithm with exact linear convergence for decentralized consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 7, pp. 346–358, 2021.

[23] B. Li, S. Cen, Y. Chen, and Y. Chi, "Communication-efficient distributed optimization in networks with gradient tracking and variance reduction," *The Journal of Machine Learning Research*, vol. 21, no. 1, 2020.

[24] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton distributed optimization methods," *IEEE Transactions on Signal Processing*, 2016.

[25] A. Maritan, G. Sharma, L. Schenato, and S. Dey, "Network-giant: Fully distributed newton-type optimization via harmonic hessian consensus," *2023 IEEE Globecom Workshops*, 2023.

[26] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of parallel and distributed computing*, vol. 67, no. 1, pp. 33–46, 2007.

[27] D. Dheeru and E. K. Taniskidou, "UCI machine learning repository."

[28] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *ATT Labs*, vol. 2, 2010.