A COMMUNICATION EFFICIENT QUASI-NEWTON METHOD FOR LARGE-SCALE DISTRIBUTED MULTI-AGENT OPTIMIZATION

Yichuan Li[1]*, Petros G. Voulgaris*[2]*, and Nikolaos M. Freris*[3]

[1]Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
[2]Department of Mechanical Engineering, University of Nevada, Reno
[3]School of Computer Science, University of Science and Technology of China

ABSTRACT

We propose a communication efficient quasi-Newton method for large-scale multi-agent convex composite optimization. We assume the setting of a network of agents that cooperatively solve a global minimization problem with strongly convex local cost functions augmented with a non-smooth convex regularizer. By introducing consensus variables, we obtain a block-diagonal Hessian and thus eliminate the need for additional communication when approximating the objective curvature information. Moreover, we reduce computational costs of existing primal-dual quasi-Newton methods from $\mathcal{O}(d^3)$ to $\mathcal{O}(cd)$ by storing c pairs of vectors of dimension d. An asynchronous implementation is presented that removes the need for coordination. Global linear convergence rate in expectation is established, and we demonstrate the merit of our algorithm numerically with real datasets.

***Index Terms*—** Distributed optimization, quasi-Newton methods, distributed learning.

1. INTRODUCTION

Distributed multi-agent optimization [1] has found numerous applications in a range of fields, such as compressed sensing [2], distributed learning [3], and sensor networks [4]. A canonical problem can be formulated as follows:

$$\underset{\hat{w}\in\mathbb{R}^d}{\text{minimize}} \left\{ \sum_{i=1}^{m} f_i(\hat{w}) + g(\hat{w}) \right\}, \tag{1}$$

where $f_i(\cdot)$ is a strongly convex local cost function pertaining to agent i and $g(\cdot)$ is a convex but possibly nonsmooth local regularizer (for example, the ℓ_1-norm that aims to promote sparsity in the solution). We assume the setting of a network of agents who cooperatively solve (1) for the common decision variable $\hat{w}$ by exchanging messages with their neighbors. First order methods [5]-[7] have been popular choices for tackling problem (1) due to their simplicity and economical computational costs. However, such methods typically feature slow convergence rate, whence a remedy is to resort to higher-order information to accelerate convergence. Newton methods use the Hessian of the objective to scale the gradient so as to accelerate the convergence speed. However, Newton methods suffer from two main drawbacks that hinder their broader applicability in distributed optimization: (i) they are limited to low dimensional problems since, at each iteration, a linear system of equations has to be solved which incurs $\mathcal{O}(d^3)$ computational costs, (ii) in multi-agent networks, the Newton step for each agent depends on its neighbors. In other words, computing the Newton step would involve multiple rounds of communication as in [8]-[11].

Contributions: (i) We propose a communication efficient quasi-Newton method based on the Alternating Direction Method of Multipliers (ADMM) [12]. We decouple agents from their neighbors through the use of consensus variables. This achieves a block-diagonal Hessian so that quasi-Newton steps can be computed locally *without additional communication among agents*. (ii) By storing c number of vector pairs, we reduce the computational costs of existing primal-dual quasi-Newton methods from $\mathcal{O}(d^3)$ to $\mathcal{O}(cd)$. (iii) We present an asynchronous implementation scheme that only involves a subset of agents at each iteration, thus removing the need for network-wide coordination. (iv) We establish global linear convergence rate in expectation without backtracking, and demonstrate its merits with numerical experiments on real datasets.

2. PRELIMINARIES

2.1. Problem formulation

The network of agents is captured by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, m\}$ is the vertex set and $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$ is the edge set, i.e., $(i, j) \in \mathcal{E}$ if and only if agent i can communicate with agent j. The total number of communication links is denoted by $n = |\mathcal{E}|$ and the neighborhood of agent i is represented as $\mathcal{N}_i := \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$. We further define the source and destination matrices $\hat{A}_s, \hat{A}_d \in \mathbb{R}^{n\times m}$ as

This work was supported by the Ministry of Science and Technology of China under grant 2019YFB2102200 and the Anhui Dept. of Science and Technology under grant 201903a05020049. Corresponding author is N. Freris.

follows: we let the k-th edge be represented by the k-th row of $\hat{A}_s$ and $\hat{A}_d$, i.e., $[\hat{A}_s]_{ki} = [\hat{A}_d]_{kj} = 1$, while all other entries in the k-th row are zero. Using the above definition, we introduce local copies of the decision variable, i.e, w_i held by agent i, and reformulate problem (1) to the consensus setting as follows:

$$\begin{aligned} \underset{w_i,\,\theta,\,z_{ij}\,\in\mathbb{R}^d}{\text{minimize}} & \left\{\sum_{i=1}^{m} f_i(w_i) + g(\theta)\right\} \\ \text{s.t.} \quad & w_i = z_{ij} = w_j \quad \forall\, i \text{ and } j \in \mathcal{N}_i, \\ & w_l = \theta \quad \text{for some } l \in [m]. \end{aligned} \tag{2}$$

We separate the decision variable corresponding to the smooth part and the nonsmooth part of the objective by enforcing an equality constraint: $w_l = \theta$, for arbitrarily chosen $l \in [m]$. Moreover, the consensus variable $\{z_{ij}\}$ decouples w_i from w_j which is *crucial* to achieve a communication-efficient implementation. Through the use of $\{z_{ij}\}$, we show that the curvature of agent i does not depend on its neighbors, whence only local information is needed to construct estimates. See Section 3 for detailed discussion. When the network graph is connected, problem (2) is equivalent to problem (1) in the sense that the optimal solution coincides, i.e., $\hat{w}^\star = w_i^\star = \theta^\star = z_{ij}^\star$ for all $i \in [m]$ and $j \in \mathcal{N}_i$. By stacking w_i, z_{ij} into column vectors $w \in \mathbb{R}^{md}, z \in \mathbb{R}^{nd}$ respectively, we define $F(w) = \sum_{i=1}^m f_i(w_i)$, whence problem (2) can be compactly expressed as:

$$\begin{aligned} \underset{\theta\in\mathbb{R}^d, w\in\mathbb{R}^{md}, z\in\mathbb{R}^{nd}}{\text{minimize}} & \Big\{F(w) + g(\theta)\Big\} \\ \text{s.t.} \quad & Aw = \begin{bmatrix}\hat{A}_s \otimes I_d \\ \hat{A}_d \otimes I_d\end{bmatrix} w = \begin{bmatrix} I_{nd} \\ I_{nd}\end{bmatrix} z = Bz, \\ & S^\top w = \theta, \end{aligned} \tag{3}$$

where $\otimes$ denotes the Kronecker product, $A \in \mathbb{R}^{2nd\times md}$ is formed by using the aforementioned source and destination matrices, and $S := (s_l \otimes I_d)$ is formed by using the vector $s_l \in \mathbb{R}^m$ whose entries are zero except the l-th entry being 1.

2.2. Introduction to L-BFGS

Quasi-Newton methods [13] constitute a popular substitute for Newton methods when the dimension d makes solving for the Newton updates computationally burdensome. In contrast to the Newton method, quasi-Newton methods do not directly invoke the Hessian but instead seek to estimate the curvature through finite differences of iterate and gradient values. In subsequent presentation, we focus on the quasi-Newton method proposed by Broyden, Fletcher, Goldfarb, and Shannon (BFGS) [14]-[15]. Specifically, we define the iterate difference $s^t := w^{t+1} - w^t$ and the gradient difference $q^t := \nabla f^{t+1} - \nabla f^t$. A Hessian inverse approximation scheme is iteratively obtained as follows:

$$\left(\widehat{H}^{t+1}\right)^{-1} = \left(V^t\right)^\top \left(\widehat{H}^t\right)^{-1} V^t + \rho^t s^t \left(s^t\right)^\top, \tag{4}$$

where $\rho^t = 1/\langle q^t, s^t\rangle$, and $V^t = I - \rho^t q^t (s^t)^\top$. In the BFGS scheme [16], the approximation $(\widehat{H}^t)^{-1}$ is stored and thus the update can be directly computed as $r^t = -(\widehat{H}^t)^{-1}\nabla f^t$ *without solving* a linear system as in the Newton method: this reduces the computation costs from $\mathcal{O}(d^3)$ to $\mathcal{O}(d^2)$ for general problems. Nevertheless, (4) approximates $(\widehat{H}^t)^{-1}$ as a function of the initialization $(\widehat{H}^0)^{-1}$ and $\{s^k, q^k\}_{k=0}^{t-1}$, i.e., the entire history of the iterates. Since early iterates tend to carry less information on the current curvature, Limited-memory BFGS (L-BFGS) [17] was proposed to store only $\{s^k, q^k\}_{k=t-c}^{t-1}$ to estimate the update direction using Algorithm 1 [13]. Note that L-BFGS operates solely on vectors, thus it not only reduces the storage costs from $\mathcal{O}(d^2)$ to $\mathcal{O}(cd)$, but also only requires $\mathcal{O}(cd)$ computation.

Algorithm 1 Two-Loop recursion of L-BFGS

Input: $h = \nabla f^t$, $\{s^k, q^k\}_{k=t-c}^{t-1}$
1: **for** $k = t-1, \ldots, t-c$ **do**
2: $\quad \alpha^k \leftarrow \rho^k \langle s^k, h\rangle$
3: $\quad h \leftarrow h - \alpha^k q^k$
4: **end for**
5: $r \leftarrow (\widehat{H}^{t,0})^{-1} h$
6: **for** $k = t-c, \ldots, t-1$ **do**
7: $\quad \beta \leftarrow \rho^k \langle q^k, r\rangle$
8: $\quad r \leftarrow r + (\alpha^k - \beta) s^k$
9: **end for**
Output: $r = (\widehat{H}^{t,c})^{-1}\nabla f^t$

3. PROPOSED METHOD

ADMM solves (3) and equivalently (1) by operating on the Augmented Lagrangian defined as:

$$\begin{aligned} \mathcal{L}(w,\theta,z;y,\lambda) = F(w) + g(\theta) + y^\top(Aw - Bz) \\ +\lambda^\top(S^\top w - \theta) + \tfrac{\mu_z}{2}\|Aw - Bz\|^2 + \tfrac{\mu_\theta}{2}\left\|S^\top w - \theta\right\|^2. \end{aligned}$$

At each iteration, ADMM sequentially minimizes $\mathcal{L}(\cdot)$ with respect to primal variables (w, θ, z) and then performs gradient ascent on dual variables (y, λ). Specifically,

$$w^{t+1} = \underset{w}{\text{argmin}}\ \mathcal{L}(w, \theta^t, z^t; y^t, \lambda^t), \tag{5a}$$

$$\theta^{t+1} = \underset{\theta}{\text{argmin}}\ \mathcal{L}(w^{t+1}, \theta, z^t; y^t, \lambda^t), \tag{5b}$$

$$z^{t+1} = \underset{z}{\text{argmin}}\ \mathcal{L}(w^{t+1}, \theta^{t+1}, z; y^t, \lambda^t), \tag{5c}$$

$$y^{t+1} = y^t + \mu_z(Aw^{t+1} - Bz^{t+1}), \tag{5d}$$

$$\lambda^{t+1} = \lambda^t + \mu_\theta(S^\top w^{t+1} - \theta^{t+1}). \tag{5e}$$

Note that above updates fall into the category of 3-block ADMM which is not guaranteed to converge for arbitrary $\mu_z, \mu_\theta > 0$ [18]. Moreover, step (5a) requires the solution of a sub-optimization problem: this typically involves multiple inner-loops and thus induces heavy computational burden on agents. We propose to approximate step (5a) by performing the following *one-step* update:

$$w^{t+1} = w^t - (\widehat{H}^t)^{-1}\nabla_w \mathcal{L}(w^t, \theta^t, z^t; y^t, \lambda^t), \tag{6}$$

where $(\widehat{H}^t)^{-1}$ is obtained by using L-BFGS (Algorithm 1) for an appropriate choice of $\{s^k, q^k\}$ pairs. Step (5b) is equivalent to the following proximal step:

$$\begin{aligned}\theta^{t+1} &= \mathbf{prox}_{g/\mu_\theta}\left(S^\top w^{t+1} + \tfrac{1}{\mu_\theta}\lambda^t\right) \\ &= \operatorname*{argmin}_\theta \left\{ g(\theta) + \tfrac{\mu_\theta}{2}\left\| S^\top w^{t+1} + \tfrac{1}{\mu_\theta}\lambda^t - \theta \right\|^2 \right\}.\end{aligned} \tag{7}$$

Remark: First note that the Hessian of $\mathcal{L}(\cdot)$ with respect to w can be expressed as:

$$H^t = \nabla^2 F(w^t) + \mu_z D + \mu_\theta S S^\top, \tag{8}$$

where $D = A^\top A \in \mathbb{R}^{md}$ is a constant diagonal matrix with its i-th block being $|\mathcal{N}_i| I_d$. Moreover, since $F(w) = \sum_{i=1}^m f_i(w_i)$, we conclude that H^t is *block diagonal*. This is only possible by introducing the intermediate consensus variable z. If the consensus constraint is directly enforced as $w_i = w_j$, then H^t would have the same sparsity pattern as the network, i.e., the ij-th block of H^t would be nonzero if $(i,j) \in \mathcal{E}$. Having a block-diagonal Hessian is *crucial* to eliminate inner-loops for computing quasi-Newton steps. This is because the presence of off-diagonal entries dictates that computing the updates would involve K additional communication rounds among the network, where $K \geq 0$ denotes the number of terms used in the Taylor expansion of $(H^t)^{-1}$ [19]. We proceed to define the $\{s_i^t, q_i^t\}$ pair for L-BFGS approximation used by the i-th agent as follows:

$$\begin{aligned} q_i^k &:= \nabla f_i(w_i^{k+1}) - \nabla f_i(w_i^k) + (\mu_z |\mathcal{N}_i| + \delta_{il}\mu_\theta + \epsilon)\, s_i^k, \\ s_i^k &:= w_i^{k+1} - w_i^k, \end{aligned} \tag{9}$$

where $\delta_{il} = 1$ if $i = l$ and 0 otherwise, and $\epsilon > 0$ brings additional robustness to the algorithm. We emphasize that q_i^k can be computed entirely using local information of agent i; this is due to the intermediate consensus variables $\{z_{ij}\}$ as explained above. Moreover, since we approximate the Hessian of the augmented Lagrangian (8) instead of just $\nabla^2 F(w^t)$, the update $(\widehat{H}^t)^{-1}\nabla_w \mathcal{L}$ can be computed at the cost of $\mathcal{O}(cd)$ using Algorithm 1. This is in contradistinction to [19] which opts to approximate $\nabla^2 F(w^t)$ only, whence a linear system has to be solved. We proceed to state a lemma that allows for efficient implementation.

Lemma 1. Define $A_s := \hat{A}_s \otimes I_d$ $A_d := \hat{A}_d \otimes I_d$. With zero initialization of $\{y^t, z^t\}$, the following holds: y^t can be decomposed as $y^t = [(\alpha^t)^\top, -(\alpha^t)^\top]^\top$,

$$\begin{aligned}\nabla_w \mathcal{L}^t = \nabla F(w^t) + \phi^t + \tfrac{\mu_z}{2} L_s w^t + \mu_\theta S(S^\top w^t - \theta^t) \\ + S\lambda^t, \text{ and } z^t = \tfrac{1}{2}(A_s + A_d) w^t.\end{aligned} \tag{10}$$

where $\phi^t := (A_s - A_d)^\top \alpha^t$ and $L_s := (A_s - A_d)^\top (A_s - A_d)$ corresponds to the graph Laplacian.

There are two implications from Lemma 1: (i) we only need to store and update half of y^t since it contains entries that are opposite of each other; (ii) there is no need to explicitly store and update z^t since it evolves on a manifold defined by w^t.

We explicate the proposed method in Algorithm 2: it admits an asynchronous implementation, where τ_i represents the counter for agent i. Each agent uses a buffer that stores the most recent $\{w_j\}$ values communicated from its neighbors. An active agent i first retrieves $w_j, j \in \mathcal{N}_i$, and then compute the corresponding subvector $\nabla_w \mathcal{L}_i^t$ in line 4. The quasi-Newton update is then computed in the line 5 at the cost of $\mathcal{O}(cd)$, where c is the number of $\{s_i^k, q_i^k\}$ pairs stored. After updating w_i (in line 6), active agent i proceeds to update ϕ_i in line 7. The l-th agent additionally performs updates pertaining to the nonsmooth regularization function (lines 9-12). The stored $\{s_i^k, q_i^k\}$ is updated by discarding the one that is computed $\tau_i - c$ steps ago when agent i was activated and adding the most recent $\{s_i^{\tau_i}, q_i^{\tau_i}\}$.

Algorithm 2 L-BFGS−ADMM

Initialization: zero initialization for all variables.

1: **for** $t = 0, \ldots, T$ **do**
2: **for** each active agent i **do**
3: Retrieve $w_j, j \in \mathcal{N}_i$ from the buffer.
4: $h_i \leftarrow \nabla f_i(w_i) + \phi_i + \frac{\mu_z}{2}\sum_{j\in\mathcal{N}_i}(w_i - w_j) + \delta_{il}\mu_\theta(w_i - \theta_i + \frac{1}{\mu_\theta}\lambda_i)$
5: Compute the update direction r_i using Algorithm 1 with h_i and $\{s_i^k, q_i^k\}_{k=\tau_i - c}^{\tau_i - 1}$ defined in (9).
6: $w_i \leftarrow w_i - r_i$
7: $\phi_i \leftarrow \phi_i + \frac{\mu_z}{2}\sum_{j\in\mathcal{N}_i}(w_i - w_j)$
8: Broadcast w_i to neighbors
9: **if** $i = l$ **then**
10: $\theta_i \leftarrow \mathbf{prox}_{g_i/\mu_\theta}\left(w_i + \frac{1}{\mu_\theta}\lambda_i\right)$
11: $\lambda_i \leftarrow \lambda_i + \mu_\theta(w_i - \theta_i)$
12: **end if**
13: Store $\{s_i^{\tau_i}, q_i^{\tau_i}\}$ and discard $\{s_i^{\tau_i - c}, q_i^{\tau_i - c}\}$
14: **end for**
15: **end for**

4. CONVERGENCE ANALYSIS

The analysis is carried assuming a connected network along with the following assumption on local cost functions.

Assumption 1. Local cost functions $f_i : \mathbb{R}^d \to \mathbb{R}, i \in [m]$, are strongly convex with constant m_f and have Lipschitz contin-

uous gradient with constant M_f. The local regularizer $g_i(\cdot) : \mathbb{R}^d \to \mathbb{R}$ is proper, closed, and convex.

We denote the unique optimal primal pair as $(w^\star, \theta^\star)$, and the unique dual pair as $(\alpha^\star, \lambda^\star)$, where $\alpha^\star$ lies in the column space of $(A_s - A_d)$. Primal uniqueness follows from strong convexity and we refer the reader to [20] for existence and uniqueness of dual optimal solutions. We proceed to state a lemma that captures the effect of replacing the exact optimization step (5a) with a quasi-Newton step (6).

Lemma 2. Define $L_u := (A_s + A_d)^\top (A_s + A_d)$ and $E_s := (A_s - A_d)$. Under Assumption 1, the iterates generated by L-BFGS−ADMM and the optimal $(w^\star, \alpha^\star, \lambda^\star)$ satisfy:

$$\nabla F(w^{t+1}) - \nabla F(w^\star) + E_s^\top(\alpha^{t+1} - \alpha^\star) + S\Big(\lambda^{t+1} - \lambda^\star + \mu_\theta(\theta^{t+1} - \theta^t)\Big) + \left(\tfrac{\mu_z}{2} L_u + \epsilon I\right)(w^{t+1} - w^t) + e^t = 0$$

where $e^t = \nabla F(w^t) - \nabla F(w^{t+1}) + (\widehat{H}^t - \mu_z D - \mu_\theta S S^\top - \epsilon I)(w^{t+1} - w^t)$. *Proof*: See the Appendix of [21].

Lemma 2 captures the error e^t induced when replacing the primal optimization step (5a) with a quasi-Newton step, i.e., $e^t = 0$ corresponds to (5a). We first introduce the following notation before presenting Theorem 1. Denote $u = [w^\top, z^\top, \theta^\top, \alpha^\top, \lambda^\top]^\top$ and similarly $u^\star$ as the optimum, $\sigma^+_{\min}$ as the smallest positive eigenvalue of $\begin{bmatrix} E_s \\ S^\top \end{bmatrix} \begin{bmatrix} E_s^\top & S \end{bmatrix}$, and $\sigma^G_{\max}, \sigma^G_{\min}$ as the largest and the smallest eigenvalue of L_u. We capture the asynchrony of the algorithm by defining Ω^t as a diagonal random matrix whose blocks activate the corresponding agent (updates w_i) or edge (updates α_i). We denote $\mathbb{E}^t[\Omega^{t+1}] = \Omega$ and $p_{\min} := \min_i p_i$, with p_i being the probability of each agent and edge being active.

Theorem 1. Recall $E_s := A_s - A_d$ and $L_u := (A_s + A_d)^\top (A_s + A_d)$ from Lemma 2. Let Assumption 1 hold and assume each agent is activated infinitely often, i,e, $p_{\min} > 0$, and let $\mu_z = 2\mu_\theta$, the iterates generated by the L-BFGS−ADMM using constant initialization γI satisfy:

$$\mathbb{E}^t\left[\left\|u^{t+1} - u^\star\right\|^2_{\mathcal{H}\Omega^{-1}}\right] \le \left(1 - \delta \tfrac{p_{\min}}{1+\delta}\right)\left\|u^t - u^\star\right\|^2_{\mathcal{H}\Omega^{-1}},$$

where $\mathcal{H} := \textbf{Blkdiag}[\epsilon I, 2\mu_z I, \mu_\theta I, \frac{2}{\mu_z} I, \frac{1}{\mu_\theta} I]$, $\tau^t \le 2\gamma + 2c(M_f + \mu_\theta(d_{\max} + 2) + \epsilon)$, $d_{\max} = \max_i |\mathcal{N}_i|$, and δ satisfy:

$$\delta = \min\left\{ \left(\tfrac{2m_f M_f}{m_f + M_f} - \tfrac{1}{\zeta}\right) \tfrac{1}{\epsilon + \mu_\theta(\sigma^{L_u}_{\max} + 2)}, \tfrac{1}{2}, \tfrac{2}{5}\tfrac{\mu_\theta \sigma^+_{\min}}{m_f + M_f}, \right.$$
$$\left. \tfrac{\mu_\theta \sigma^+_{\min}(\epsilon - \zeta(\tau^t)^2)}{5((\tau^t)^2 + \epsilon^2)}, \tfrac{\sigma^+_{\min}}{5\max\{1, \sigma^{L_u}_{\max}\}} \right\}. \quad (11)$$

Proof: See the Appendix of [21].

5. EXPERIMENTS

We evaluate the proposed L-BFGS−ADMM against existing distributed algorithms for multi-agent convex composite optimization problems, namely P2D2 [7] and PG-EXTRA [6].

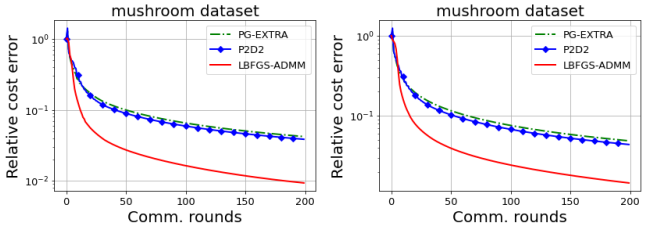

Fig. 1: Performance comparison on the mushrooms dataset with dimension $d = 112$. The network size for the left figure is $m = 10$ and for the right $m = 20$. Each agent of L-BFGS−ADMM stores $c = 10$ pairs of $\{s_i, q_i\}$. All algorithms are synchronous.

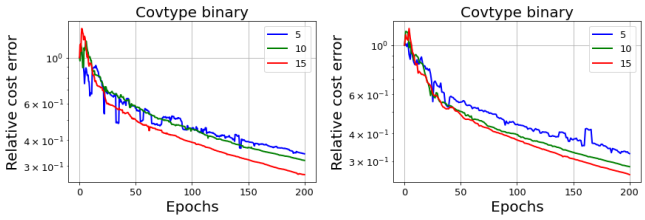

Fig. 2: Effects of number of copies stored and the number of activated agents. We assume a network of 10 agents and activate 2 agents and 5 agents uniformly at random at the left and right plots respectively. We use the same hyperparameters for all cases with c = 5, 10, 15.

We do not compare against PD-QN [19] since it does not support nonsmooth regularizers and requires $\mathcal{O}(d^3)$ computation costs. We consider the following problem:

$$\underset{w \in \mathbb{R}^d}{\text{minimize}}\ \mathcal{J}(w) = \left\{ \frac{1}{m} \sum_{i=1}^{m} f_i(w) + \rho g(w) \right\},$$

where $f_i(w) = \frac{1}{m_i} \sum_{j=1}^{m_i} \left[\ln\left(1 + e^{w^\top x_j}\right) + (1 - y_j) w^\top x_j \right]$, $g(w) = \|w\|_1$, and $\rho = 0.0005$. We denote m_i as the number of data points held by agent i, where each data point contains a feature-label pair $(x_j, y_j) \in \mathbb{R}^d \times \{0, 1\}$. We initialize $(\widehat{H}^{t,0})^{-1}$ as a diagonal matrix $\Gamma^t I$, where the i-th block is given by $\gamma_i^t = \frac{(s_i^{t-1})^\top q_i^{t-1}}{(q_i^{t-1})^\top q_i^{t-1}}$. Such initialization aims to estimate the norm of the Hessian along the most recent update direction [13]. We take 5,000 data points from the covtype dataset with dimension $d = 54$ and the mushrooms dataset with dimension $d = 112$ respectively. Both datasets are available from the UCI Machine Learning Repository. We plot the averaged relative costs error $\mathrm{RE}(t) := \frac{\frac{1}{m}\sum_{i=1}^m \mathcal{J}(w_i^t) - \mathcal{J}(w^\star)}{\frac{1}{m}\sum_{i=1}^m \mathcal{J}(w_i^0) - \mathcal{J}(w^\star)}$ in all cases. From Fig 1, we observe that the proposed algorithm outperforms both baseline methods using the number of communication rounds as the metric. On the other hand, Fig. 2 shows that storing more copies of $\{s_i, q_i\}$ can effectively reduce oscillations and achieve faster convergence rate.

6. REFERENCES

[1] Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl H. Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019.

[2] Haixiao Liu, Bin Song, Hao Qin, and Zhiliang Qiu, "An Adaptive-ADMM algorithm with support and signal value detection for compressed sensing," *IEEE Signal Processing Letters*, vol. 20, pp. 315–318, 2013.

[3] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer, "A survey on distributed machine learning," *ACM Computing Survey*, vol. 53, 2020.

[4] N. M. Freris, H. Kowshik, and P. R. Kumar, "Fundamentals of large sensor networks: Connectivity, capacity, clocks, and computation," *Proceedings of the IEEE*, pp. 1828–1846, 2010.

[5] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, pp. 48–61, 2009.

[6] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Transactions on Signal Processing*, vol. 63, pp. 6013–6023, 2015.

[7] S. Alghunaim, K. Yuan, and A. H. Sayed, "A linearly convergent proximal gradient algorithm for decentralized optimization," in *Advances in Neural Information Processing Systems 32*, pp. 2848–2858. 2019.

[8] Aryan Mokhtari, Qing Ling, and Alejandro Ribeiro, "Network Newton distributed optimization methods," *IEEE Transactions on Signal Processing*, vol. 65, pp. 146–161, 2017.

[9] Y. Li, N. M. Freris, P. Voulgaris, and D. Stipanović, "D-SOP: Distributed second order proximal method for convex composite optimization," in *Proceedings of the American Control Conference*, pp. 2844–2849. 2020.

[10] Fatemeh Mansoori and Ermin Wei, "A fast distributed asynchronous Newton-based optimization algorithm," *IEEE Transactions on Automatic Control*, vol. 65, pp. 2769–2784, 2020.

[11] Yichuan Li, Nikolaos M. Freris, Petros Voulgaris, and Dušan Stipanović, "DN-ADMM: Distributed newton ADMM for multi-agent optimization," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 3343–3348.

[12] Stephen Boyd and Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[13] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 2006.

[14] William C. Davidon, "Variable metric method for minimization," *SIAM Journal on Optimization*, vol. 1, pp. 1–17, 1991.

[15] Donald Goldfarb, "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, pp. 23–26, 1970.

[16] Yichuan Li, Yonghai Gong, Nikolaos M. Freris, Petros Voulgaris, and Dušan Stipanović, "BFGS-ADMM for large-scale distributed optimization," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 1689–1694.

[17] Dong C. Liu and Jorge Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, p. 503–528, 1989.

[18] Tianyi Lin, Shiqian Ma, and Shuzhong Zhang, "Global convergence of unmodified 3-block ADMM for a class of convex minimization problems," *Journal of Scientific Computing*, vol. 76, pp. 69–88, 2018.

[19] Mark Eisen, Aryan Mokhtari, and Alejandro Ribeiro, "A primal-dual quasi-Newton method for exact consensus optimization," *IEEE Transactions on Signal Processing*, vol. 67, pp. 5983–5997, 2019.

[20] 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, pp. 1750–1761, 2014.

[21] Yichuan Li, Petros G Voulgaris, and Nikolaos M Freris, "A communication efficient quasi-newton method for large-scale distributed multi-agent optimization," *arXiv preprint arXiv:2201.03759*, 2022.