FedHybrid: A Hybrid Federated Optimization Method for Heterogeneous Clients

Xiaochun Niu and Ermin Wei, *Member, IEEE*

***Abstract*—We consider a distributed consensus optimization problem over a server-client (federated) network, where all clients are connected to a central server. Current distributed algorithms fail to capture the heterogeneity in clients' local computation capacities. Motivated by the method of multipliers in centralized optimization, we derive a Newton-type primal-dual method with a distributed implementation utilizing the server-client topology. Then we propose *FedHybrid* as a hybrid primal-dual method that allows heterogeneous clients to perform different types of updates. Specifically, those clients with higher computational capabilities and/or cheaper costs to perform computation can implement Newton-type updates locally, while other clients can adopt much simpler gradient-type updates. Theoretically, we propose a novel merit function by combining the dual optimality gap and the primal tracking error. We prove that FedHybrid converges linearly to the exact optimal point for strongly convex functions, regardless of clients' choices of gradient-type or Newton-type updates. Finally, we show numerical studies to demonstrate the efficacy of our method in practice. To the best of our knowledge, this is the first hybrid method allowing heterogeneous local updates for distributed consensus optimization with provable convergence and rate guarantees.**

***Index Terms*—Server-client networks, distributed optimization, primal-dual algorithm, heterogeneous systems.**

I. Introduction

We study the distributed optimization problem over a *server-client (federated) network,*

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

where n is the number of clients in the network connected to a central node (server) and $f_i : \mathbb{R}^d \to \mathbb{R}$ is the local objective function corresponding to the ith client. In such problems, each client only has access to its local data and communicates with the central server. The goal of the system is to learn a shared model over all the data, represented by ω, in the network without exchanging local data due to privacy issues or communication limitations [1]. For instance, in a supervised learning setting, if we consider an empirical risk minimization problem, the local objective function f_i represents expected loss over the local data distribution at the ith client.

Solving Problem 1 using a distributed procedure over the server-client network where both data collection and model training is pushed to a massive number of edge clients has gained significant attention recently. This is motivated by a wide range of applications such as multi-vehicle and multi-robot networks [2], [3], machine learning [4], and especially *federated learning* [5], [6].

To develop a distributed algorithm for solving Problem 1, we decouple the computation of individual clients by introducing local copies of the decision variable. We index the central server as the 0th node and denote by $x_0, x_i \in \mathbb{R}^d$ local copies of ω kept at the server and client i, respectively. We formulate Problem I under the server-client network as a *consensus optimization* problem [7], [8],

$$\min_{x_0,x_1,\ldots,x_n} \sum_{i=1}^{n} f_i(x_i) \ \text{ s.t. } x_0 = x_i, \ \text{for } i \in \{1,\ldots,n\}. \tag{2}$$

We remark that the consensus constraint $x_0 = x_i$ for $i \in \{1,\ldots,n\}$ enforces the equivalence of Problems 1 and 2.

In practice, distributed systems, particularly federated setups [1], [9], may involve heterogeneous clients [10] due to the drastically varying storage, computation, and communication capabilities among the clients caused by hardware, network connectivity, and battery power. This work focuses on heterogeneous systems where clients have various computation capabilities. Namely, some can efficiently compute second-order information, whereas others can only process first-order information. Such systems are common nowadays since processors with advanced computation capabilities have limited availability due to the recent global chip shortage. Consequently, many distributed computation systems have only a few agents with advanced hardware co-existing with many older processors. While there is proliferating literature on developing distributed methods to solve Problem 2, most existing works consider homogeneous settings where all clients implement the same kind of local computation, such as all gradient-type methods [8], [11], [12] or all Newton-type methods [13], [14], [15], [16]. In this paradigm of algorithmic development, a fast-converging method utilizing higher-order information may not be applicable to a distributed system if any agent cannot handle high-order

Manuscript received 24 February 2022; revised 19 October 2022 and 19 December 2022; accepted 19 January 2023. Date of publication 26 January 2023; date of current version 14 February 2023. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Kobi Cohen. This work was supported by the National Science Foundation (NSF) under Grants ECCS-2030251 and CMMI-2024774. *(Corresponding author: Xiaochun Niu.)*

Xiaochun Niu is with the Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208 USA (e-mail: xiaochunniu2024@u.northwestern.edu).

Ermin Wei is with the Department of Electrical and Computer Engineering, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208 USA (e-mail: ermin.wei@northwestern.edu).

Digital Object Identifier 10.1109/TSP.2023.3240083

computation. The resulting system could not fully utilize the distributed computation capability. It is, therefore, imperative to provide a flexible and efficient hybrid method to utilize heterogeneous clients. To the best of our knowledge, this paper takes the first step in this direction.

To utilize the heterogeneous system, we propose a distributed hybrid method for the server-client (federated) network named *FedHybrid*, which allows some clients to perform gradient-type updates while others use Newton-type information. The goal is to maximally utilize varying local computation capabilities to speed up the system's performance and improve overall communication efficiency. Specifically, we first introduce a gradient-type primal-dual method with a practical distributed implementation. Moreover, motivated by the superlinear convergence of Newton-type primal-dual methods in centralized settings, we propose a Newton-type distributed method as a distributed approximation of the centralized method to improve convergence speed. In particular, we form a block diagonal approximation of the dual Hessian utilizing the server-client topology. Then we propose FedHybrid as a combination of gradient-type and Newton-type primal-dual methods. Specifically, those clients with higher computational capabilities and/or cheaper cost to perform computation can implement Newton-type updates locally, while other clients can adopt much simpler gradient-type updates. To analyze the intricate coupling of primal and dual updates in FedHybrid, we combine the dual optimality gap and the primal tracking error as a novel merit function, based on which we show that FedHybrid achieves a linear (Q-linear) convergence rate to the exact optimal solution for strongly convex functions. Numerical experiments on both synthetic and real-life data are conducted to demonstrate the efficacy of our method.

We clarify that we name the Federated Hybrid (FedHybrid) method mainly due to the server-client topology of the system. The method serves as a starting point for hybrid methods combining first and second-order updates to utilize heterogeneous computation capabilities among clients, a crucial issue frequently arising in federated settings, and ignores some practical requirements of federated learning, including partial participation of clients, multiple local updates, and nonconvex objectives. We consider these practical requirements as future directions with modifications and variants of FedHybrid.

A. Literature Review

Our work is related to the growing literature on distributed algorithms for solving Problem 2. A line of works studies the general network topology. First-order primal iterative methods, including distributed (sub)-gradient descent (DGD) and related methods [8], [11], [17], have updates in the form of linear combinations of a local gradient descent step and a weighted average among local neighbors. Second-order primal methods, including Network Newton [18] and Distributed Newton method [19], rely on iterative schemes to approximate a Newton step. In work [20], the authors derive a DGD-based method with the inclusion of first and second-order updates in the continuous-time setting. Their method cannot be directly applied in discrete time and lacks convergence rate analysis. In addition, the method NEXT proposed in [21] combines successive convex approximation techniques with consensus mechanisms while also lacking a rate analysis. Another popular approach is to use dual decomposition-based methods such as ADMM [22], [23], ESOM [24], and CoCoA [25]. Among these, ESOM performs second-order updates in the primal space and first-order updates in the dual space and has a provable linear convergence rate.

Another line of research focuses on the server-client network, also known as the federated learning/optimization setting. Existing works include primal first-order methods like FedAvg [5], [26], FedProx [27], FedSplit [28], FedAC [29], FedNova [30], and adaptive federated methods [31]; and primal-dual methods like FedPD [32] that utilizes first-order updates in both the primal and the dual spaces. Such methods suffer from slow convergence speed due to their first-order nature. Some second-order methods have been proposed. Examples include DANE [13], DiSCO [14], GIANT [15], DINGO [16], and DAve-QN [33]. However, all of these methods except for GIANT require an inner loop to approximate Newton's direction at each iteration. Also, DANE, DiSCO, and GIANT only give the best performance when each client has access to i.i.d. local data, which is not realistic in practice, especially in federated optimization settings. Moreover, none of the existing works for either general topology or server-client architecture allow heterogeneous client updates.

B. Contributions

Our main contributions are fourfold:

- We propose FedHybrid as a heterogeneity-enabling primal-dual method, where clients in the network can perform different types (gradient or Newton-type) of updates simultaneously based on their computation capacities.
- If all clients choose second-order updates, we develop a Newton-type primal-dual method in the distributed scheme by approximating both the primal and the dual Hessian matrices utilizing the server-client network.
- We propose a novel merit function and show that FedHybrid converges to the exact optimal solution in a linear (Q-linear) rate for strongly convex functions, independent of clients' choice of first or second-order updates.
- We provide numerical experiments to demonstrate the efficacy of FedHybrid in practice.

To the best of our knowledge, FedHybrid is the first hybrid method allowing heterogeneous local updates for federated optimization with a provable convergence and rate guarantee. FedHybrid significantly enhances the flexibility and applicability of distributed methods to various hardware applications.

Notations: For any $m \in \mathbb{Z}^+$, we denote by I_m the identity matrix, $\mathbb{1}_m = (1, \ldots, 1)^\intercal \in \mathbb{R}^m$, and $[m] = \{1, \ldots, m\}$. Let $\otimes$ denote the Kronecker product. For any symmetric positive definite matrix $X \in \mathbb{R}^{p\times p}$, we denote by $\theta_{\min}(X)$ its smallest eigenvalue and $\|y\|_X^2 = y^\intercal X y$ the quadratic form for any y.

II. Preliminaries

In this section, we formulate Problem 2 in a compact form and review the method of multipliers in centralized optimization.

The method of multipliers is derived by formulating a dual problem based on the augmented Lagrangian function [34], which motivates our derivation of FedHybrid.

We first assume the following conditions for local functions.

Assumption 1: For any agent $i \in [n]$, the local function f_i is twice differentiable and m_i-strongly convex and its gradient ∇f_i is ℓ_i-Lipschitz continuous with constants $0 < m_i \leq \ell_i$.

For any agent $i \in [n]$, Assumption 1 implies that eigenvalues of the local Hessian are bounded by $m_i I_d \preceq \nabla^2 f_i(\cdot) \preceq \ell_i I_d$. We define $m = \min_{i\in[n]}\{m_i\}$ and $\ell = \max_{i\in[n]}\{\ell_i\}$.

Problem Reformulation: Problem 2 has the compact form,

$$\min_{\widetilde{x}\in\mathbb{R}^{(n+1)d}} \quad \widetilde{f}(\widetilde{x}) = \sum_{i=1}^{n} f_i(x_i) \quad \text{s.t. } W\widetilde{x} = 0, \tag{3}$$

where $x = (x_1^\intercal, \ldots, x_n^\intercal)^\intercal$ and $\widetilde{x} = (x_0^\intercal, x^\intercal)^\intercal$ are the concatenations of local variables, $\widetilde{f} : \mathbb{R}^{(n+1)d} \to \mathbb{R}$ is the aggregate function, and $W = (\mathbb{1}_n, -I_n) \otimes I_d \in \mathbb{R}^{nd\times(n+1)d}$ is the incidence matrix of the server-client network. We remark that the ith row of $W\widetilde{x} = 0$ represents the consensus constraint $x_0 = x_i$ in Problem 2. Since $\widetilde{f}(\widetilde{x})$ only depends on the value of x, for convenience, we also define $f : \mathbb{R}^{nd} \to \mathbb{R}$, $f(x) = \sum_{i=1}^n f_i(x_i) = \widetilde{f}(\widetilde{x})$. We differentiate whether a variable includes the server's decision or clients' by the tilde.

Dual Problem: Now we introduce the dual problem of Problem 3 based on the augmented Lagrangian. We denote by $\lambda = (\lambda_1^\intercal, \ldots, \lambda_n^\intercal)^\intercal$ the dual variable with $\lambda_i \in \mathbb{R}^d$ associated with the ith constraint $x_0 = x_i$ and define the augmented Lagrangian function $\widetilde{L}(\widetilde{x}, \lambda)$ of Problem 3 as follows,

$$\widetilde{L}(\widetilde{x}, \lambda) = \widetilde{f}(\widetilde{x}) + \lambda^\intercal W\widetilde{x} + \frac{\mu}{2}\widetilde{x}^\intercal W^\intercal W\widetilde{x}, \tag{4}$$

where $\mu > 0$ is a constant. We remark that the matrix $W^\intercal W$ is the graph Laplacian of the network. The augmented Lagrangian in (4) can also be viewed as the (unaugmented) Lagrangian associated with the penalized problem

$$\min_{\widetilde{x}\in\mathbb{R}^{(n+1)d}} \widetilde{f}(\widetilde{x}) + \frac{\mu}{2}\widetilde{x}^\intercal W^\intercal W\widetilde{x} \quad \text{s.t. } W\widetilde{x} = 0. \tag{5}$$

Problem 5 is equivalent to Problem 3 since the augmentation term $\mu\widetilde{x}^\intercal W^\intercal W\widetilde{x}/2$ is zero for any feasible $\widetilde{x}$. Otherwise, the term serves as a penalty for the violation of the consensus constraint. By Assumption 1 and the Slater's condition, strong duality holds for Problem 5 [35]. Thus, Problem 5, as well as Problem 3, is equivalent to the following dual problem,

$$\max_{\lambda\in\mathbb{R}^{nd}} g(\lambda), \quad \text{where } g(\lambda) = \min_{\widetilde{x}\in\mathbb{R}^{(n+1)d}} \widetilde{L}(\widetilde{x}, \lambda), \tag{6}$$

where we refer to $g : \mathbb{R}^{nd} \to \mathbb{R}$ as the dual function. For any $\lambda \in \mathbb{R}^{nd}$, as we will show in Lemma 8, $\widetilde{L}(\cdot, \lambda)$ is a strongly convex function with a unique minimizer. We define

$$\widetilde{x}^*(\lambda) = \operatorname*{argmin}_{\widetilde{x}\in\mathbb{R}^{(n+1)d}} \widetilde{L}(\widetilde{x}, \lambda), \tag{7}$$

the unique minimizer of the inner problem in Problem 6. By the definition of g in (6), we have $L(\widetilde{x}^*(\lambda), \lambda) = g(\lambda)$ for any $\lambda \in \mathbb{R}^{nd}$. As we will prove in Lemma 9, the dual function g is strongly concave with a unique maximizer, which we denote by λ^*. We define $\widetilde{x}^{\mathsf{OPT}}$ as the optimal solution of Problem 3. Then the strong duality implies that $\widetilde{x}^*(\lambda^*) = \widetilde{x}^{\mathsf{OPT}}$.

Method of Multipliers (MM): We now review MM [34], [36], which solves Problem 3 in the dual space based on the augmented Lagrangian $\widetilde{L}$ and helps to motivate our derivation of FedHybrid. At each iteration k, MM finds the exact primal minimizer $\widetilde{x}^*(\lambda^k)$ and take one step in the dual space

$$\begin{aligned} \widetilde{x}^*(\lambda^k) &= \operatorname*{argmin}_{\widetilde{x}} \widetilde{L}(\widetilde{x}, \lambda^k), \\ \lambda^{k+1} &= U(\widetilde{x}^*(\lambda^k), \lambda^k), \end{aligned} \tag{8}$$

where $U : \mathbb{R}^{(n+1)d\times nd} \to \mathbb{R}$ is a general dual update formula with the property that $\lambda^* = U(\widetilde{x}^*(\lambda^*), \lambda^*)$. We give some popular choices for U as follows,

$$U_1(\widetilde{x}^*(\lambda^k), \lambda^k) = \lambda^k + \beta_1\nabla g(\lambda^k), \tag{9a}$$

$$U_2(\widetilde{x}^*(\lambda^k), \lambda^k) = \lambda^k - \beta_2\Delta\lambda^k, \tag{9b}$$

where $\beta_1, \beta_2 > 0$ are stepsizes and $\Delta\lambda^k$ is the Newton's direction satisfying $\nabla^2 g(\lambda^k)\Delta\lambda^k = \nabla g(\lambda^k)$. We remark that (9a) and (9b) correspond to the gradient ascent method and the Newton's method with respect to the dual function $g(\lambda)$, respectively. The following lemma shows the explicit forms of $\nabla g(\lambda)$ and $\nabla^2 g(\lambda)$ used in (9a) and (9b).

Lemma 2 (Dual Gradient and Hessian [36]): Under Assumption 1, with $\widetilde{x}^*(\lambda)$ defined in (7), the gradient and the Hessian of the dual function $g(\lambda)$ defined in Problem 6 are given by

$$\nabla g(\lambda) = W\widetilde{x}^*(\lambda), \ \nabla^2 g(\lambda) = -W(\nabla^2_{\widetilde{x}\widetilde{x}}\widetilde{L}(\widetilde{x}^*(\lambda), \lambda))^{-1}W^\intercal.$$

We remark that the primal update in (8) is computationally expensive due to the requirement of an exact solution to the inner minimization problem and it cannot be readily implemented in a distributed manner due to the nonseparable augmentation term $\mu\widetilde{x}^\intercal W^\intercal W\widetilde{x}/2$ in $\widetilde{L}(\widetilde{x}, \lambda)$.

III. Algorithm

In this section, we first introduce a distributed gradient-type primal-dual method to reduce the computation complexity of MM, which is a particular case of FedHybrid with all gradient-type clients. For a speedup, we derive a Newton-type primal-dual method utilizing the server-client topology, which leads to another particular case of FedHybrid with all Newton-type clients. Finally, we combine them to propose FedHybrid, which allows a mixture of first and second-order updates by various clients and provides flexibility to handle and utilize heterogeneity in the network. In all of these methods, the clients could only process local information and are not allowed to communicate local Hessian matrices due to the expensive communication costs and privacy concerns.

A. Gradient-Type Primal-Dual Method

To develop a first-order method based on the dual gradient ascent update in (9a), we need to compute $\nabla g(\lambda^k) = W\widetilde{x}^*(\lambda)$ by Lemma 2. However, the computation of the exact $\widetilde{x}^*(\lambda^k)$ can be computationally expensive. Thus, we approximate it by

taking a gradient descent step in the primal space, leading to the gradient-type primal-dual algorithm. At each iteration k,

$$\widetilde{x}^{k+1} = \widetilde{x}^k - \alpha_1 \nabla_{\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k),$$
$$\lambda^{k+1} = \lambda^k + \beta_1 W \widetilde{x}^k, \tag{10}$$

where $\alpha_1, \beta_1 > 0$ are the primal and the dual stepsizes, respectively. This recovers the Arrow-Hurwicz-Uzawa method [37]. The updates in (10) can be easily implemented in a distributed scheme as follows. At each iteration k, each client $i \in [n]$ performs a primal and a dual updates,

$$x_i^{k+1} = x_i^k - \alpha_1(\nabla f_i(x_i^k) - \lambda_i^k + \mu(x_i^k - x_0^k)),$$
$$\lambda_i^{k+1} = \lambda_i^k + \beta_1(x_0^k - x_i^k); \tag{11}$$

and the central server updates by aggregating information,

$$x_0^{k+1} = x_0^k - \alpha_1 \left[\sum_{i=1}^n \lambda_i^k + \mu \left(n x_0^k - \sum_{i=1}^n x_i^k \right) \right]. \tag{12}$$

This is a special case of FedHybrid when all clients perform gradient-type updates. While this gradient-type primal-dual method leads to simple distributed implementation, it suffers from slow convergence due to its first-order nature. This motivates us to consider Newton's method for a speedup.

B. Newton-Type Primal-Dual Method

We now derive a Newton-type primal-dual method utilizing the server-client network topology.

Primal Update: We consider a Newton's step as an approximation of the primal update in (8). The primal Hessian $\nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k) = \nabla^2 \widetilde{f}(\widetilde{x}^k) + \mu W^\intercal W$ is nonseparable due to the graph Laplacian $W^\intercal W$, which makes it intractable to compute the exact Hessian inverse in a decentralized manner. Thus, we replace $W^\intercal W$ by its block diagonal $E = \mathrm{diag}\{n, 1, \ldots, 1\} \otimes I_d \in \mathbb{R}^{(n+1)d \times (n+1)d}$. By using $\widetilde{H}^k = \nabla^2 \widetilde{f}(\widetilde{x}^k) + \mu E$ as an approximation of the primal Hessian $\nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k)$, we obtain the following Newton-type primal update. At each iteration k,

$$\widetilde{x}^{k+1} = \widetilde{x}^k - (\widetilde{H}^k)^{-1} \nabla_{\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k). \tag{13}$$

In particular, the server update takes the following form,

$$x_0^{k+1} = \frac{1}{n} \sum_{i \in [n]} x_i^k - \frac{1}{\mu n} \sum_{i \in [n]} \lambda_i^k. \tag{14}$$

We remark that when taking $\alpha_1 = 1/(\mu n)$ in (12), the central server will have the same updates as in (14).

Dual Update: We consider the exact dual Newton update $\Delta\lambda^k$ in (9b). We note that it's difficult to compute $\Delta\lambda^k$ in a distributed manner since the dual Hessian $\nabla^2 g(\lambda^k)$ given in Lemma 2 is nonseparable. Thus, in order to obtain a Newton-type dual update in a distributed scheme, we will provide an estimator of $\Delta\lambda^k$. We first substitute $\widetilde{x}^k$ as an approximation of $\widetilde{x}^*(\lambda^k)$ and define $\widehat{\nabla} g(\lambda^k)$ and $\widehat{\nabla}^2 g(\lambda^k)$ as estimators of $\nabla g(\lambda^k)$ and $\nabla^2 g(\lambda^k)$, respectively, as follows,

$$\widehat{\nabla} g(\lambda^k) = W \widetilde{x}^k, \ \widehat{\nabla}^2 g(\lambda^k) = -W(\nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k))^{-1} W^\intercal.$$

We define a Hessian weighted average of primal variables as,

$$y^k = \left[\sum_{i \in [n]} \nabla^2 f_i(x_i^k) \right]^{-1} \left[\sum_{i \in [n]} \nabla^2 f_i(x_i^k) x_i^k \right]. \tag{15}$$

We introduce the following lemma to characterize the estimator $\Delta\widehat{\lambda}^k$ of $\Delta\lambda^k$, where $\Delta\widehat{\lambda}^k$ satisfies,

$$\widehat{\nabla}^2 g(\lambda^k) \Delta\widehat{\lambda}^k = \widehat{\nabla} g(\lambda^k). \tag{16}$$

Lemma 3 (Approximated Dual Newton Update): Under Assumption 1, with the incidence matrix W and y^k defined in (15), the dual Newton update $\Delta\widehat{\lambda}^k$ in (16) satisfies

$$W^\intercal \Delta\widehat{\lambda}^k = \nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k)(\mathbb{1}_{n+1} \otimes y^k - \widetilde{x}^k).$$

Proof: By the formula in (16) and the dual gradient $\widehat{\nabla} g(\lambda^k)$ and the dual Hessian $\widehat{\nabla}^2 g(\lambda^k)$ given in Lemma 2, we have

$$-W(\nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k))^{-1} W^\intercal \Delta\widehat{\lambda}^k = W \widetilde{x}^k.$$

We note that the null space of the matrix W is $\mathrm{Null}(W) = \{\mathbb{1}_{n+1} \otimes y : y \in \mathbb{R}^d\}$. Thus, there exists $y^k \in \mathbb{R}^d$ such that

$$(\nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k))^{-1} W^\intercal \Delta\widehat{\lambda}^k + \widetilde{x}^k = \mathbb{1}_{n+1} \otimes y^k.$$

Rearranging terms in the previous equation, we have

$$W^\intercal \Delta\widehat{\lambda}^k = \nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k)(\mathbb{1}_{n+1} \otimes y^k - \widetilde{x}^k)$$
$$= (\nabla^2 \widetilde{f}(\widetilde{x}^k) + \mu W^\intercal W)(\mathbb{1}_{n+1} \otimes y^k - \widetilde{x}^k), \tag{17}$$

where the last equality follows from the explicit form of $\nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k)$. Since $(\mathbb{1}_{n+1}^\intercal \otimes I_d) W^\intercal = 0$, by multiplying $\mathbb{1}_{n+1}^\intercal \otimes I_d$ on both sides of (17), we have

$$0 = (\mathbb{1}_{n+1}^\intercal \otimes I_d) \nabla^2 \widetilde{f}(\widetilde{x}^k)(\mathbb{1}_{n+1} \otimes y^k - \widetilde{x}^k).$$

Then we have $\left(\sum_{i \in [n]} \nabla^2 f_i(x_i^k) \right) y^k = \sum_{i \in [n]} \nabla^2 f_i(x_i^k) x_i^k$, and thus, $y^k = \left[\sum_{i \in [n]} \nabla^2 f_i(x_i^k) \right]^{-1} \left[\sum_{i \in [n]} \nabla^2 f_i(x_i^k) x_i^k \right]$. Substituting the preceding relation into (17), we have

$$W^\intercal \Delta\widehat{\lambda}^k = \nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k)(\mathbb{1}_{n+1} \otimes y^k - \widetilde{x}^k),$$

where y^k is defined above. □

We remark that calculating y^k in Lemma 3 directly is impractical in a distributed way, since communicating $d \times d$ local Hessian matrices can be prohibitively expensive. We then introduce another approximation of estimating y^k using x_0^k in (14). The next lemma justifies this approximation.

Lemma 4 (First-Order Stationary Condition): At the optimal point of Problem 6, it holds that $\sum_{i \in [n]} \lambda_i^* = 0$.

Proof: For the optimal primal-dual pair $(\widetilde{x}^*, \lambda^*)$, based on optimality and feasibility, we have $\nabla_{\widetilde{x}} \widetilde{L}(\widetilde{x}, \lambda) \,|_{(\widetilde{x}^*, \lambda^*)} = 0$ and $W \widetilde{x}^* = 0$. This implies that $\nabla \widetilde{f}(\widetilde{x}^*) + W^\intercal \lambda^* = 0$. Thus, the part related to the central variable x_0 can be written as $\mathbb{1}_n^\intercal \lambda^* = 0$. □

The above lemma implies that x_0^k in (14) can be viewed as a penalized average of primal variables $\{x_i^{k-1}\}_{i \in [n]}$, where the penalty term vanishes at the optimal point, i.e., $x_0^* =$

$\sum_{i\in[n]} x_0^*/n - \sum_{i\in[n]} \lambda_i^*/(\mu n) = x_0^*$. Moreover, due to the consensus constraint, $x_i^* = x_0^*$ for any $i \in [n]$, we have $y^* = (\sum_{i\in[n]} \nabla^2 f_i(x_i^*))^{-1} \sum_{i\in[n]} \nabla^2 f_i(x_i^*) x_0^* = x_0^*$. Thus, at the optimal point, the estimator x_0 is equal to y.

If we substitute x_0^k defined in (14) and $\widetilde{H}^k$ defined above as estimators of y^k and $\nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k)$ in Lemma 3, respectively, then we obtain an estimator $\Delta\breve{\lambda}^k$ of $\Delta\widehat{\lambda}^k$ satisfying $W^\intercal \Delta\breve{\lambda}^k = \widetilde{H}^k(\mathbb{1}_{n+1} \otimes x_0^k - \widetilde{x}^k)$. Then by the structure of the incidence matrix W, we have

$$\Delta\breve{\lambda}^k = -H^k W \widetilde{x}^k, \tag{18}$$

where $H^k = \nabla^2 f(x^k) + \mu I_{nd} \in \mathbb{R}^{nd\times nd}$ is a submatrix of $\widetilde{H}^k$, corresponding to components related to the clients. Thus, using the dual Newton's formula in (9b) with the estimator defined in (18), we obtain a Newton-type dual update

$$\lambda^{k+1} = \lambda^k + \beta_2 H^k W \widetilde{x}^k. \tag{19}$$

By combining (13) and (19), we obtain a Newton-type primal-dual method as follows. At each iteration k,

$$\begin{aligned} \widetilde{x}^{k+1} &= \widetilde{x}^k - (\widetilde{H}^k)^{-1} \nabla_{\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k), \\ \lambda^{k+1} &= \lambda^k + \beta_2 H^k W \widetilde{x}^k, \end{aligned} \tag{20}$$

where $H^k = \nabla^2 f(x^k) + \mu I_{nd}$ and $\widetilde{H}^k = \mathrm{diag}\{\mu n I_d, H^k\}$. This is a special case of FedHybrid when all clients perform Newton-type updates. The following is the distributed implementation of (20). At iteration k, each client $i \in [n]$ performs

$$\begin{aligned} x_i^{k+1} &= x_i^k \\ &\quad - (\nabla^2 f_i(x_i) + \mu I_d)^{-1}(\nabla f_i(x_i^k) - \lambda_i^k + \mu(x_i^k - x_0^k)), \\ \lambda_i^{k+1} &= \lambda_i^k + \beta_2(\nabla^2 f_i(x_i) + \mu I_d)(x_0^k - x_i^k); \end{aligned} \tag{21}$$

and the server aggregates information and updates using (14).

C. FedHybrid to Handle System Heterogeneity

Since the clients in the network may be heterogeneous, we consider combining gradient-type method in (10) and Newton-type method in (20) to provide a hybrid update framework. Specifically, all clients in the network can choose to perform gradient-type or Newton-type updates based on their computation capabilities. For notational convenience, we denote $J_1 = \{i \in [n] : \text{client i performs gradient-type updates}\}$ and similarly, $J_2 = \{i \in [n] : \text{client i performs Newton-type updates}\}$. Based on such choices of different update types, we propose our hybrid updates as follow. At each iteration k, we have

$$\begin{aligned} \widetilde{x}^{k+1} &= \widetilde{x}^k - \widetilde{A}(\widetilde{D}^k)^{-1} \nabla_{\widetilde{x}} \widetilde{L}(\widetilde{x}^k, \lambda^k), \\ \lambda^{k+1} &= \lambda^k + B D^k W \widetilde{x}^k, \end{aligned} \tag{22}$$

where $\widetilde{A} = \mathrm{diag}\{a_0, a_1, \ldots, a_n\} \otimes I_d \in \mathbb{R}^{(n+1)d\times(n+1)d}$ and $B = \mathrm{diag}\{b_1, \ldots, b_n\} \otimes I_d \in \mathbb{R}^{nd\times nd}$ with personalized stepsizes $a_i, b_i > 0$ and update matrices $\widetilde{D}^k = \mathrm{diag}\{I_d, D^k\} \in \mathbb{R}^{(n+1)d\times(n+1)d}$ and $D^k = \mathrm{diag}\{D_1^k, \ldots, D_n^k\} \in \mathbb{R}^{nd\times nd}$. Here $D_i^k = I_d$ if $i \in J_1$ while $D_i^k = \nabla^2 f_i(x_i) + \mu I_d$ if $i \in J_2$.

Algorithm 1: FedHybrid: Hybrid Primal-dual Algorithm for distributed consensus optimization.

1: **Input:** Initialization index sets J_1 and J_2, x_0^0, x_i^0, $\lambda_i^0 \in \mathbb{R}^d$ and $a_i, b_i \in \mathbb{R}^+$ for all $i \in [n]$, and penalty parameter μ.
2: **for** $k = 1, \ldots, K-1$ **do**
3: Server sends x_0^k to all clients in the network;
4: **for** each client $i \in J_1$ **do** ▷Gradient-type
5: $x_i^{k+1} = x_i^k - a_i(\nabla f_i(x_i^k) - \lambda_i^k + \mu(x_i^k - x_0^k))$;
6: $\lambda_i^{k+1} = \lambda_i^k + b_i(x_0^k - x_i^k)$;
7: Send x_i^{k+1} and λ_i^{k+1} to the server;
8: **end for**
9: **for** each client $i \in J_2$ **do** ▷Newton-type
10: $x_i^{k+1} = x_i^k - a_i(\nabla^2 f_i(x_i^k) + \mu I_d)^{-1}(\nabla f_i(x_i^k) - \lambda_i^k + \mu(x_i^k - x_0^k))$;
11: $\lambda_i^{k+1} = \lambda_i^k + b_i(\nabla^2 f_i(x_i^k) + \mu I_d)(x_0^k - x_i^k)$;
12: Send x_i^{k+1} and λ_i^{k+1} to the server;
13: **end for**
14: Server consensus updates: $x_0^{k+1} = (\sum_{i\in[n]} x_i^{k+1})/n - (\sum_{i\in[n]} \lambda_i^{k+1})/(\mu n)$.
15: **end for**

The updates in (22) generalize both (10) and (20). On the one extreme, if $J_1 = [n]$ and $J_2 = \varnothing$, (22) recovers the gradient-type updates in (10); on the other extreme, if $J_1 = \varnothing$ and $J_2 = [n]$, (22) recovers the Newton-type updates in (20). Using (22), we propose the FedHybrid method with a distributed implementation based on (11) and (21) in Algorithm 1.

FedHybrid in Algorithm 1 consists of two steps: gradient-type or Newton-type updates at the clients, simultaneously, followed by a consensus update at the server. Specifically, gradient-type (Lines 4–8) and Newton-type updates (Lines 9–13) follow from (22) by extracting the corresponding block. For the consensus update, to simplify the method, we choose the stepsize $a_0 = 1/(\mu n)$ in $\widetilde{A}$ of (22) and replace the primal and dual decision variables x^k and λ^k by their updated counterpart x^{k+1} and λ^{k+1}. By substitution, we obtain that

$$x_0^{k+1} = \frac{1}{n}\sum_{i\in[n]} x_i^{k+1} - \frac{1}{\mu n}\sum_{i\in[n]} \lambda_i^{k+1}, \tag{23}$$

which corresponds to the consensus update in Line 14.

D. Discussions

Naive Attempt: A naive attempt to solve Problem 2 using a mixture of first and second-order information would be $x_i^{k+1} = x_i^k - \nabla f_i(x_0^k)$ for i in J_1 and $x_i^{k+1} = x_i^k - [\nabla^2 f_i(x_0^k)]^{-1} \nabla f_i(x_0^k)$ for i in J_2 with $x_0^{k+1} = \sum_{i\in[n]} x_i^{k+1}$. The server's update is equivalently written as $x_0^{k+1} = x_0^k - \sum_{i\in J_2} \nabla f_i(x_0^k) - \sum_{i\in J_2} (\nabla^2 f_i(x_0^k)^{-1} \nabla f_i(x_0^k))$. However, at the optimal solution, we have $\sum_{i\in[n]} \nabla f_i(x^*) = 0$, and the Hessian weighted local gradients do not sum to 0. Therefore this iteration does not converge to x^*. This was the reason behind our usage of the Lagrangian formulation.

Multiple Primal Updates: FedAvg ($x_i^{k+1} = x_i^k - a_i \nabla f_i(x_0^k)$ and $x_0^{k+1} = \sum_{i\in[n]} x_i^{k+1}$) with more than one local update does not go to the correct optimal point [28], whereas [38] shows the exact convergence to the optimal solution of primal-dual gradient method with any number of local primal updates. Thus, Lagrangian-based methods are more likely to work with multiple local updates. This is desirable in federated learning settings, as multiple local updates can save communication costs. We leave this as a promising open direction to explore.

Connections with Other Primal-Dual Methods: FedHybrid runs one primal and one dual step for each agent at each iteration instead of local rounds of x_i-updates to solve the primal problem to a certain accuracy, as required by FedProx [27], FedSplit [28], and FedPD [32]. Though Algorithm 1 requires all clients' synchronous participation and exact local gradient/Hessian computations, we consider asynchronous updates (partial participation) and stochastic gradient or subsampled Newton [39] local updates as important future directions. We note while full client participation is not always possible, it occurs in certain engineered systems, such as autonomous vehicles platoon [40] and high-performance-computing clusters [41].

Heterogeneity in Computation Capabilities: We highlight that FedHybrid in Algorithm 1 is designed to utilize the system heterogeneity in computation capabilities among agents. Namely, some agents can process second-order information efficiently, while others can only compute first-order information. On the other hand, another notion of system heterogeneity often considered in distributed settings is the dissimilarities of local objective functions (datasets). Due to its primal-dual update framework, we remark that FedHybrid also handles the latter scenario compared with some primal-only methods like FedAvg with multiple local updates and FedProx. The analysis and implementation of FedHybrid do not require any similarity assumptions on local functions.

IV. Theoretical Convergence

This section presents the linear convergence rate for FedHybrid in Algorithm 1, regardless of clients' choices of gradient-type or Newton-type updates. In Section IV-A, we reformulate the augmented Lagrangian $\widetilde{L}$ and obtain a simplified representation of FedHybrid. In Section IV-B, we show strong convexity/concavity and Lipschitz smoothness of the primal and the dual functions. Finally, in Section IV-C, we propose a novel merit function and show that FedHybrid converges to the exact optimal solution at a linear rate.

A. Reformulation Based on the Consensus Update.

In this section, we reformulate the augmented Lagrangian $\widetilde{L}$ based on the consensus update (23) and obtain a function L. For convenience, we will use L in the subsequent analysis.

We note that the consensus update (23) in Algorithm 1 can be written as $x_0^{k+1} = X_0(x^{k+1}, \lambda^{k+1})$, where $X_0 : \mathbb{R}^{nd} \times \mathbb{R}^{nd} \to \mathbb{R}$ is a function defined as follows,

$$X_0(x,\lambda) = (\mathbb{1}_n^\intercal \otimes I_d)[x/n - \lambda/(\mu n)]. \tag{24}$$

By substituting $x_0 = X_0(x,\lambda)$ in (24) into the consensus constraint in Problem 3, we have

$$W\widetilde{x} = \mathbb{1}_n \otimes X_0(x,\lambda) - x = -Mx - Z\lambda/\mu, \tag{25}$$

where matrices $Z = (\mathbb{1}_n\mathbb{1}_n^\intercal) \otimes I_d/n \in \mathbb{R}^{nd\times nd}$ and $M = I_{nd} - Z$. It's easy to show that both matrices are idempotent, that is, $Z^2 = Z$ and $M^2 = M$. Thus, we have $\|Z\| = \|M\| = 1$. Moreover, if we substitute $x_0 = X_0(x,\lambda)$ into the augmented Lagrangian $\widetilde{L}$ defined in (4) and define $L : \mathbb{R}^{nd} \times \mathbb{R}^{nd} \to \mathbb{R}$ such that $L(x,\lambda) = \widetilde{L}(X_0(x,\lambda), x, \lambda)$, then we have

$$L(x,\lambda) = f(x) - \lambda^\intercal Mx + \frac{\mu}{2}x^\intercal Mx - \frac{1}{2\mu}\lambda^\intercal Z\lambda. \tag{26}$$

The following lemmas studies the connection between L and $\widetilde{L}$, which enables a simplified representation of FedHybrid.

Lemma 5 (Partial Gradients of $\widetilde{L}$ and L): Under Assumption 1, with $x_0 = X_0(x,\lambda)$, for any $x, \lambda \in \mathbb{R}^{nd}$, it holds that

$$\nabla_x L(x,\lambda) = \nabla_x \widetilde{L}(x_0, x, \lambda)\,|_{\,x_0 = X_0(x,\lambda)}.$$

By the definition of X_0, we have $\nabla_x L(x,\lambda) = \nabla_x \widetilde{L}(x_0,x,\lambda)\,|_{\,x_0=X_0(x,\lambda)} = \nabla f(x) - M\lambda + \mu Mx$. By Lemma 5, the primal update in Algorithm 1 is as follows,

$$\begin{aligned} x^{k+1} &= x^k - A(D^k)^{-1}\nabla_x \widetilde{L}(x_0^k, x^k, \lambda^k)\,|_{\,x_0^k = X_0(x^k,\lambda^k)} \\ &= x^k - A(D^k)^{-1}\nabla_x L(x^k,\lambda^k). \end{aligned}$$

Thus, we can rewrite FedHybrid in Algorithm 1 in a compact form. At each iteration k, FedHybrid consists of the steps,

$$\begin{aligned} &x^{k+1} = x^k - A(D^k)^{-1}\nabla_x L(x^k,\lambda^k), \\ &\lambda^{k+1} = \lambda^k + BD^k W\widetilde{x}^k, \quad x_0^{k+1} = X_0(x^{k+1},\lambda^{k+1}), \end{aligned} \tag{27}$$

where $W\widetilde{x}^k$ is defined in (25). For convenience, we will use the equivalent representation (27) of FedHybrid in the subsequent analysis. Next, we characterize the primal optimal point $\widetilde{x}^*(\lambda)$ for the inner problem in Problem 6.

Lemma 6 (Inner Primal Optimal Point): Given any $\lambda \in \mathbb{R}^{nd}$, we consider the primal optimal point $\widetilde{x}^*(\lambda) = (x_0^*(\lambda)^\intercal, x^*(\lambda)^\intercal)^\intercal$ for the inner problem in (6). It holds that,

$$x_0^*(\lambda) = X_0(x^*(\lambda),\lambda).$$

Lemma 6 shows $L(x^*(\lambda),\lambda) = \widetilde{L}(X_0(x^*(\lambda),\lambda), x^*(\lambda),\lambda) = \widetilde{L}(x_0^*(\lambda), x^*(\lambda),\lambda) = \widetilde{L}(\widetilde{x}^*(\lambda),\lambda) = g(\lambda)$. Moreover, Lemmas 5 and 6 imply $\nabla_x L(x^*(\lambda),\lambda) = \nabla_x \widetilde{L}(X_0(x^*(\lambda),\lambda), x^*(\lambda),\lambda) = \nabla_x \widetilde{L}(x_0^*(\lambda), x^*(\lambda),\lambda) = \nabla_x \widetilde{L}(\widetilde{x}^*(\lambda),\lambda) = 0$, which ensures $x^*(\lambda) = \operatorname{argmin}_x L(x,\lambda)$.

B. Properties of Primal and Dual Functions.

In this section, we show the strong convexity/concavity and the Lipschitz smoothness of the primal function $L(\cdot,\lambda)$ and $\widetilde{L}(\cdot,\lambda)$ and the dual function g, respectively.

Lemma 7 (Strongly Convex and Lipschitz Smooth Primal): Under Assumption 1, for any fixed $\lambda \in \mathbb{R}^{nd}$, the function $L(\cdot,\lambda)$

is m-strongly convex and its partial gradient $\nabla_x L(x,\lambda)$ is ℓ_L-Lipschitz continuous with $\ell_L = \ell + \mu$.

Proof: Given any fixed $\lambda \in \mathbb{R}^{nd}$, by taking partial Hessian with respect to x of the function L defined in (26), we have $\nabla^2_{xx} L(x,\lambda) = \nabla^2 f(x) + \mu M$. We prove the Lemma by bounding the partial Hessian. Under Assumption 1, using the fact that $0 \preceq M \preceq I_{nd}$, we have $mI_{nd} \preceq \nabla^2_{xx} L(x,\lambda) \preceq (\ell + \mu) I_{nd}$. This concludes the proof of the Lemma. □

We also show that the original $\widetilde{L}(\cdot,\lambda)$ is strongly convex.

Lemma 8 (Strongly Convex Primal $\widetilde{L}(\cdot,\lambda)$): Under Assumption 1, for any $\lambda \in \mathbb{R}^{nd}$, the function $\widetilde{L}(\cdot,\lambda)$ is strongly convex.

Proof: We denote by $\underline{H} = \mathrm{diag}\{0, m, \ldots, m\} \otimes I_d$. Given any fixed $\lambda \in \mathbb{R}^{nd}$, by taking partial Hessian with respect to $\widetilde{x}$ of the function $\widetilde{L}$ defined in (4), we have

$$\nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x},\lambda) = \nabla^2 \widetilde{f}(\widetilde{x}) + \mu W^{\intercal} W \succeq \underline{H} + \mu W^{\intercal} W = \begin{pmatrix} \mu n I_d & -\mu \mathbb{1}_n^{\intercal} \otimes I_d \\ -\mu \mathbb{1}_n \otimes I_d & (\mu + m) I_{nd} \end{pmatrix},$$

where the inequality is due to Assumption 1 and the last equality is due to the definition of W. By the Schur complement, since $\mu n I_d \succ 0$ and $\mu n I_d - (-\mu \mathbb{1}_n^{\intercal} \otimes I_d)(\mu + m)^{-1} I_{nd} (-\mu \mathbb{1}_n \otimes I_d) = \mu n (1 - \mu/(\mu + m)) I_d \succ 0$, the matrix $\underline{H} + \mu W^{\intercal} W$ is positive definite. Thus, the matrix $\nabla^2_{\widetilde{x}\widetilde{x}} \widetilde{L}(\widetilde{x},\lambda)$ is positive definite with a uniform lower bound $\underline{H} + \mu W^{\intercal} W$ for all $\widetilde{x} \in \mathbb{R}^{(n+1)d}$. Thus, $\widetilde{L}(\cdot,\lambda)$ is strongly convex. □

Now we study the properties of the dual function $g(\lambda)$.

Lemma 9 (Strongly Concave and Lipschitz Smooth Dual): Under Assumption 1, the dual function $g(\cdot)$ is m_g-strongly concave and its gradient $\nabla g(\cdot)$ is ℓ_g-Lipschitz continuous with constants $m_g = 1/(\mu + \ell)$ and $\ell_g = 1/\mu$.

Proof: By Lemma 2, the dual Hessian is given by $\nabla^2 g(\lambda) = -W(\nabla^2 \widetilde{f}(\widetilde{x}^*(\lambda)) + \mu W^{\intercal} W)^{-1} W^{\intercal}$. We show both lower and upper bounds on $\nabla^2 g$. We define $\underline{H} = \mathrm{diag}\{0, m, \ldots, m\} \otimes I_d$ and $\overline{H} = \mathrm{diag}\{0, \ell, \ldots, \ell\} \otimes I_d \in \mathbb{R}^{(n+1)d \times (n+1)d}$. Under Assumption 1, we have $(\overline{H} + \mu W^{\intercal} W)^{-1} \preceq (\nabla^2 \widetilde{f}(\widetilde{x}^*(\lambda)) + \mu W^{\intercal} W)^{-1} \preceq (\underline{H} + \mu W^{\intercal} W)^{-1}$ for any $\lambda \in \mathbb{R}^{nd}$.

For any constant $s > 0$, we consider the matrix $S = \mathrm{diag}\{0, s, \ldots, s\} \otimes I_d \in \mathbb{R}^{(n+1)d \times (n+1)d}$. By the inverse of a block matrix using Schur complement [35], we have

$$(S + \mu W^{\intercal} W)^{-1} = \begin{pmatrix} \mu n I_d & -\mu \mathbb{1}_n^{\intercal} \otimes I_d \\ -\mu \mathbb{1}_n \otimes I_d & (\mu + s) I_{nd} \end{pmatrix}^{-1} = \begin{pmatrix} \frac{\mu + s}{\mu n s} I_d & \frac{1}{ns} \mathbb{1}_n^{\intercal} \otimes I_d \\ \frac{1}{ns} \mathbb{1}_n \otimes I_d & \frac{1}{\mu + s} I_{nd} + \frac{\mu}{s(\mu + s)} Z \end{pmatrix}.$$

Then by straightforward matrix multiplications, we have

$$W(S + \mu W^{\intercal} W)^{-1} W^{\intercal} = I_{nd}/(\mu + s) + sZ/[\mu(\mu + s)].$$

Thus, using the fact that $0 \preceq Z \preceq I$, we have

$$I_{nd}/(\mu + s) \preceq W(S + \mu W^{\intercal} W)^{-1} W^{\intercal} \preceq I_{nd}/\mu.$$

We also note that for any matrix $S \in \mathbb{R}^{(n+1)d \times (n+1)d}$, if $\underline{S} \preceq S \preceq \overline{S}$, we have $W \underline{S} W^{\intercal} \preceq W S W^{\intercal} \preceq W \overline{S} W^{\intercal}$. Thus, we obtain a lower and an upper bounds on the dual Hessian,

$$I_{nd}/(\mu + \ell) \preceq W(\overline{H} + \mu W^{\intercal} W)^{-1} W^{\intercal} \preceq -\nabla^2 g(\lambda) \preceq W(\underline{H} + \mu W^{\intercal} W)^{-1} W^{\intercal} \preceq I_{nd}/\mu.$$

Therefore, we conclude the proof of the Lemma. □

We remark that there are existing literature [42], [43], [44] studying dual problems and showing the strong concavity and Lipschitz gradients of the dual function g. While here we provide tighter bounds with constants m_g and ℓ_g defined in Lemma 9 utilizing the structure of the server-client topology.

C. Convergence Analysis of FedHybrid.

In this section, we show the convergence analysis of FedHybrid. We first introduce the performance metrics.

Merit Function: We define $z = (x; y)$. Most existing analyses of primal-dual gradient methods study $\|z^k - z^*\|$ as the merit function [45], [46], which decreases geometrically such that $\|z^{k+1} - z^*\|_V^2 \leq \rho \|z^k - z^*\|_V^2$ for a matrix $V \succeq 0$ and a constant $\rho > 0$ implying linear convergence. However, such analysis does not apply to FedHybrid. A similar procedure leads to $\|z^{k+1} - z^*\|_{V^k}^2 \leq \rho \|z^k - z^*\|_{V^k}^2$ with time-varying $\{V^k \succeq 0\}_k$ due to the time-varying local Hessian matrices, which does not guarantee a linear rate. Thus, we customize a novel merit function and introduce a new line of analysis that combines primal function tracking error and dual function optimality gap to address the challenge of the time-varying local Hessians. We define the primal tracking error and the dual optimality gap as performance metrics,

$$\begin{aligned} \Delta_x^k &= L(x^k, \lambda^k) - L(x^*(\lambda^k), \lambda^k), \\ \Delta_\lambda^k &= g(\lambda^*) - g(\lambda^k), \end{aligned} \tag{28}$$

where $x^*(\lambda) = \mathrm{argmin}_x L(x,\lambda)$ and λ^* is the optimal solution to Problem 6. Here Δ_x^k quantifies how close the augmented Lagrangian at x^k is from the optimal value of the inner problem given λ^k and Δ_λ^k measures the distance of the current dual function value to the optimal one. We remark that since $L(x^*(\lambda^k), \lambda^k) = \widetilde{L}(\widetilde{x}^*(\lambda^k), \lambda^k)$ and $L(x^k, \lambda^k) = \widetilde{L}(\widetilde{x}^k, \lambda^k)$, we also have $\Delta_x^k = \widetilde{L}(\widetilde{x}^*(\lambda^k), \lambda^k) - \widetilde{L}(\widetilde{x}^k, \lambda^k)$. We define the merit function by combing the performance metrics in (28) as

$$\Delta^k = 13 \Delta_\lambda^k + \Delta_x^k. \tag{29}$$

We remark that by the definition of $x^*(\lambda)$ and λ^*, both Δ_x^k and Δ_λ^k are nonnegative, so does the merit function Δ^k. Thus, the convergence of Δ^k to zero guarantees the convergence of Δ_x^k and Δ_λ^k. Specifically, under Assumption 1, the convergence of Δ_x^k and Δ_λ^k ensures that the primal variable sequence $\{\widetilde{x}^k\}$ and the dual sequence $\{\lambda^k\}$ converge to the optimal $\widetilde{x}^*(\lambda^*)$ and λ^*, respectively, where $\widetilde{x}^*(\lambda^*) = \widetilde{x}^{\mathsf{OPT}}$ is the optimal solution of the original problem in (3) due to strong duality.

Linear Convergence of FedHybrid: Now we present the theoretical analysis and the linear convergence result of FedHybrid. We decompose our analysis into three steps. Due to the coupled nature of primal and dual updates in the algorithm in 27, our main idea for analyzing primal-dual methods is to first upper bound the dual optimality gap Δ_λ^k and the primal tracking error $\Delta_{\widetilde{x}}^k$ through coupled inequalities. Propositions 12 and 13 show the coupled inequalities obtained by the first two steps. Then we combines these results and show the linear convergence rate of FedHybrid in Theorem 14.

Before presenting the analysis, we introduce the following lemma, which is a corollary of the strong convexity of $L(\cdot,\lambda)$.

Lemma 10: Under Assumption 1, for all $k=0,1,\ldots,K-1$, the iterates generated from FedHybrid in Algorithm 1 satisfy $\|W\widetilde{x}^k - W\widetilde{x}^*(\lambda^k)\| \le \|\nabla_x L(x^k,\lambda^k)\|/m$.

We also provide the following lemma that bounds the dual update $\|\lambda^{k+1}-\lambda^k\|$ by an alternative primal tracking error $\|\nabla_x L(x^k,\lambda^k)\|^2$ and the dual gradient $\|\nabla g(\lambda^k)\|$. For convenience, we define a constant β as follows,

$$\beta = \max\{\max_{i\in J_1}\{b_i\}, \max_{i\in J_2}\{b_i(\ell_i+\mu)\}\}. \tag{30}$$

The constant β serves as an upper bound of $\|BD^k\|$ since

$$\|BD^k\| = \max_{i\in[n]}\{b_i\|D_i^k\|\} \le \beta. \tag{31}$$

Lemma 11: Under Assumption 1, for all $k=0,1,\ldots,K-1$, the iterates generated from FedHybrid in Algorithm 1 satisfy $\|W\widetilde{x}^k\|^2_{BD^k} \le 2\beta\|\nabla_x L(x^k,\lambda^k)\|^2/m^2 + 2\|\nabla g(\lambda^k)\|^2_{BD^k}$ and $\|\lambda^{k+1}-\lambda^k\|^2 \le 2\beta^2\|\nabla_x L(x^k,\lambda^k)\|^2/m^2 + 2\beta\|\nabla g(\lambda^k)\|^2_{BD^k}$.

Now we start the convergence analysis of FedHybrid. First, we derive a bound of the dual optimality gap Δ_λ^{k+1} with an alternative primal tracking error $\|\nabla_x L(x^k,\lambda^k)\|^2$ using the Lipschitz smoothness of the dual function g.

Proposition 12 (Bounding the Dual Optimality Gap): Under Assumption 1, given a constant $\mu>0$ and stepsizes $\{a_i,b_i\}_{i\in[n]}>0$, with β and ℓ_g defined in (30) and Lemma 9, for all $k=0,\ldots,K-1$, the iterates from Algorithm 1 satisfy

$$\Delta_\lambda^{k+1} \le \Delta_\lambda^k - \left(\frac{1}{2}-\beta\ell_g\right)\|\nabla g(\lambda^k)\|^2_{BD^k} + \left(\frac{1}{2}+\beta\ell_g\right)\frac{\beta}{m^2}\|\nabla_x L(x^k,\lambda^k)\|^2.$$

Proof: The ℓ_g-Lipschitz continuity of ∇g in Lemma 9 implies,

$$\begin{aligned} &g(\lambda^{k+1}) \\ &\ge g(\lambda^k) + \langle \nabla g(\lambda^k), \lambda^{k+1}-\lambda^k\rangle - \frac{\ell_g}{2}\|\lambda^{k+1}-\lambda^k\|^2 \\ &= g(\lambda^k) + \langle \nabla g(\lambda^k), BD^k W\widetilde{x}^k\rangle - \frac{\ell_g}{2}\|\lambda^{k+1}-\lambda^k\|^2, \end{aligned} \tag{32}$$

where the equality is due to the dual update in (27). Now we consider the second term in (32). By adding and subtracting $\langle \nabla g(\lambda^k), BD^k \nabla g(\lambda^k)\rangle$ in the inner product, we have

$$\begin{aligned} &\langle \nabla g(\lambda^k), BD^k W\widetilde{x}^k\rangle \\ &= \langle \nabla g(\lambda^k), BD^k W\widetilde{x}^k - BD^k\nabla g(\lambda^k)\rangle \\ &\quad + \langle \nabla g(\lambda^k), BD^k \nabla g(\lambda^k)\rangle \\ &\ge -\frac{1}{2}\|\nabla g(\lambda^k)\|^2_{BD^k} - \frac{1}{2}\|W\widetilde{x}^k - \nabla g(\lambda^k)\|^2_{BD^k} \\ &\quad + \|\nabla g(\lambda^k)\|^2_{BD^k} \\ &= \frac{1}{2}\|\nabla g(\lambda^k)\|^2_{BD^k} - \frac{1}{2}\|W\widetilde{x}^k - W\widetilde{x}^*(\lambda^k)\|^2_{BD^k}, \\ &\ge \frac{1}{2}\|\nabla g(\lambda^k)\|^2_{BD^k} - \frac{\beta}{2m^2}\|\nabla_x L(x^k,\lambda^k)\|^2, \end{aligned} \tag{33}$$

where the first inequality follows from the inequality that for any $a,b\in\mathbb{R}^{nd}$, $2a^\intercal b \le \|a\|^2+\|b\|^2$, the last equality is due to Lemma 2, and the last equality is due to Lemma 10.

By substituting (33) and Lemma 11 into (32), we have

$$\begin{aligned} g(\lambda^{k+1}) \ge g(\lambda^k) &+ \left(\frac{1}{2}-\beta\ell_g\right)\|\nabla g(\lambda^k)\|^2_{BD^k} \\ &- \left(\frac{1}{2}+\beta\ell_g\right)\frac{\beta}{m^2}\|\nabla_x L(x^k,\lambda^k)\|^2. \end{aligned}$$

By subtracting the optimal value $g(\lambda^*)$ and taking negative signs on both sides, we conclude the proof. □

Next, we provide an upper bound of Δ_x^{k+1} using the dual optimality gap Δ_λ^k. Specifically, we upper bound the updated primal tracking error by the Lipschitz smoothness of $L(x,\lambda)$ with respect to x and obtain the following result.

Proposition 13 (Bounding the Primal Tracking Error): Under Assumption 1, given a constant $\mu>0$ and stepsizes $\{a_i,b_i\}_{i\in[n]}>0$, for all $k=0,1,\ldots,K-1$, the iterates generated from FedHybrid in Algorithm 1 satisfy

$$\begin{aligned} \Delta_x^{k+1} \le \Delta_x^k + \kappa\|\nabla g(\lambda^k)\|^2_{BD^k} - \|\nabla_x L(x^k,\lambda^k)\|^2_{P^k} \\ + \Delta_\lambda^k - \Delta_\lambda^{k+1}, \end{aligned}$$

where constant $\kappa = 3+2\beta^2/\mu^2+\beta/\mu$ and matrix $P^k = A(D^k)^{-1} - (\beta+\ell_L/2)A^2(D^k)^{-2} - \beta\kappa/m^2 I$.

Proof: By the definition in (28), the tracking error of the primal update Δ_x^{k+1} consists of the following three terms,

$$\begin{aligned} \Delta_x^{k+1} &= L(x^{k+1},\lambda^{k+1}) - L(x^*(\lambda^{k+1}),\lambda^{k+1}) \\ &= \underbrace{L(x^{k+1},\lambda^{k+1}) - L(x^{k+1},\lambda^k)}_{\text{term (A)}} \\ &\quad + \underbrace{L(x^{k+1},\lambda^k) - L(x^*(\lambda^k),\lambda^k)}_{\text{term (B)}} \\ &\quad + \underbrace{L(x^*(\lambda^k),\lambda^k) - L(x^*(\lambda^{k+1}),\lambda^{k+1})}_{\text{term (C)}}, \end{aligned} \tag{34}$$

where term (A) measures the increase due to the dual update, term (B) represents the updated primal tracking error, and term (C) shows the difference between dual optimality gaps. In the sequel, we upper bound terms (A)–(C), respectively.

Term (A): By the definition of L in (26), we have

$$\begin{aligned} &L(x^{k+1},\lambda^{k+1}) - L(x^{k+1},\lambda^k) \\ &= (\lambda^{k+1}-\lambda^k)^\intercal(-Mx^{k+1}) - \frac{1}{2\mu}(\lambda^{k+1})^\intercal Z\lambda^{k+1} \\ &\quad + \frac{1}{2\mu}(\lambda^k)^\intercal Z\lambda^k \\ &= (\lambda^{k+1}-\lambda^k)^\intercal\left(-Mx^{k+1} - \frac{1}{\mu}Z\lambda^{k+1}\right) + \frac{1}{2\mu}(\lambda^k)^\intercal Z\lambda^k \\ &\quad + \frac{1}{\mu}(\lambda^{k+1}-\lambda^k)^\intercal Z\lambda^{k+1} - \frac{1}{2\mu}(\lambda^{k+1})^\intercal Z\lambda^{k+1} \\ &= \underbrace{(\lambda^{k+1}-\lambda^k)^\intercal W\widetilde{x}^{k+1}}_{\text{term (A.1)}} + \frac{1}{2\mu}\underbrace{(\lambda^{k+1}-\lambda^k)^\intercal Z(\lambda^{k+1}-\lambda^k)}_{\text{term (A.2)}}, \end{aligned} \tag{35}$$

where the last equality follows from (25). Next, we will bound terms (A.1) and (A.2), respectively.

Term (A.1): Based on the dual update in (27), we have

$$\begin{aligned}(\lambda^{k+1}-\lambda^k)^\intercal W\widetilde{x}^{k+1} &= (BD^kW\widetilde{x}^k)^\intercal W\widetilde{x}^{k+1}\\ &= \|W\widetilde{x}^k\|^2_{BD^k} + (W\widetilde{x}^k)^\intercal BD^k(W\widetilde{x}^{k+1}-W\widetilde{x}^k)\\ &\le \frac{3}{2}\|W\widetilde{x}^k\|^2_{BD^k} + \frac{1}{2}\|W\widetilde{x}^{k+1}-W\widetilde{x}^k\|^2_{BD^k}, \end{aligned} \tag{36}$$

where the inequality follows from the inequality that for any $a,b\in\mathbb{R}^{nd}$, $2a^\intercal b\le\|a\|^2+\|b\|^2$. We note that the first term in (36) can be upper bounded by Lemma 11. Now we upper bound the second term. Based on (25) and (27), we have

$$\begin{aligned}&\|W\widetilde{x}^{k+1}-W\widetilde{x}^k\|^2_{BD^k}\\ &= \left\|-M(x^{k+1}-x^k)-\frac{1}{\mu}Z(\lambda^{k+1}-\lambda^k)\right\|^2_{BD^k}\\ &\le 2\left\|M(x^{k+1}-x^k)\right\|^2_{BD^k} + 2\left\|\frac{1}{\mu}Z(\lambda^{k+1}-\lambda^k)\right\|^2_{BD^k}\\ &\le 2\beta\|x^{k+1}-x^k\|^2 + \frac{2\beta}{\mu^2}\|\lambda^{k+1}-\lambda^k\|^2, \end{aligned} \tag{37}$$

where the first inequality follows from the inequality that for any $a,b\in\mathbb{R}^{nd}$, $2a^\intercal b\le\|a\|^2+\|b\|^2$ and the last inequality follows from the fact that $\|M\|=\|Z\|=1$ and $\|BD^k\|\le\beta$ in (31). By the primal update in (27), we have

$$\|x^{k+1}-x^k\|^2 = \|A(D^k)^{-1}\nabla_x L(x^k,\lambda^k)\|^2. \tag{38}$$

Thus, substituting (38) and Lemma 11 into (37), we have

$$\begin{aligned}\|W\widetilde{x}^{k+1}-W\widetilde{x}^k\|^2_{BD^k} &\le 2\beta\|A(D^k)^{-1}\nabla_x L(\widetilde{x}^k,\lambda^k)\|^2\\ &+\frac{4\beta^2}{\mu^2}\|\nabla g(\lambda^k)\|^2_{BD^k} + \frac{4\beta^3}{m^2\mu^2}\|\nabla_x L(x^k,\lambda^k)\|^2. \end{aligned} \tag{39}$$

Finally, substituting Lemma 11 and (39) into (36), we have

$$\begin{aligned}(\lambda^{k+1}-\lambda^k)^\intercal W\widetilde{x}^{k+1} &\le \beta\|A(D^k)^{-1}\nabla_x L(x^k,\lambda^k)\|^2\\ &+\left(3+\frac{2\beta^2}{\mu^2}\right)\|\nabla g(\lambda^k)\|^2_{BD^k}\\ &+\frac{\beta(3\mu^2+2\beta^2)}{m^2\mu^2}\|\nabla_x L(x^k,\lambda^k)\|^2. \end{aligned} \tag{40}$$

Term (A.2): By the fact that $\|Z\|=1$ and Lemma 11, we have

$$\begin{aligned}(\lambda^{k+1}-\lambda^k)^\intercal Z(\lambda^{k+1}-\lambda^k) &\le \|Z\|\|\lambda^{k+1}-\lambda^k\|^2\\ &\le \frac{2\beta^2}{m^2}\|\nabla_x L(x^k,\lambda^k)\|^2 + 2\beta\|\nabla g(\lambda^k)\|^2_{BD^k}. \end{aligned} \tag{41}$$

Therefore, substituting (40) and (41) into (35), we have

$$\begin{aligned}&L(x^{k+1},\lambda^{k+1})-L(x^{k+1},\lambda^k)\\ &\le \frac{\beta(3\mu^2+2\beta^2+\beta\mu)}{m^2\mu^2}\|\nabla_x L(x^k,\lambda^k)\|^2\\ &+\left(3+\frac{2\beta^2}{\mu^2}+\frac{\beta}{\mu}\right)\|\nabla g(\lambda^k)\|^2_{BD^k}\\ &+\beta\|A(D^k)^{-1}\nabla_x L(\widetilde{x}^k,\lambda^k)\|^2. \end{aligned} \tag{42}$$

This provides an upper bound on term (A).

Term (B): Using the ℓ_L-Lipschitz continuity of $\nabla_x L(x,\lambda^k)$ from Lemma 5, we have

$$\begin{aligned}L(x^{k+1},\lambda^k) &\le L(x^k,\lambda^k)+\nabla_x L(x^k,\lambda^k)^\intercal(x^{k+1}-x^k)\\ &+\frac{\ell_L}{2}\|x^{k+1}-x^k\|^2\\ &= L(x^k,\lambda^k)+\frac{\ell_L}{2}\|A(D^k)^{-1}\nabla_x L(x^k,\lambda^k)\|^2\\ &-\nabla_x L(x^k,\lambda^k)^\intercal A(D^k)^{-1}\nabla_x L(x^k,\lambda^k), \end{aligned} \tag{43}$$

where the equality follows from the primal update in (27). Subtracting $L(x^*(\lambda^k),\lambda^k)$ on both sides of (43), we have the following upper bound on term (B):

$$\begin{aligned}L(x^{k+1},\lambda^k)-L(x^*(\lambda^k),\lambda^k) &\le \Delta_x^k - \|\nabla_x L(x^k,\lambda^k)\|^2_{A(D^k)^{-1}}\\ &+\frac{\ell_L}{2}\|A(D^k)^{-1}\nabla_x L(x^k,\lambda^k)\|^2. \end{aligned} \tag{44}$$

Term (C): Since for any λ, $L(x^*(\lambda),\lambda)=g(\lambda)$, we have

$$\begin{aligned}L(x^*(\lambda^k),\lambda^k)-L(x^*(\lambda^{k+1}),\lambda^{k+1}) &= g(\lambda^k)-g(\lambda^{k+1})\\ &= \Delta_\lambda^k-\Delta_\lambda^{k+1}. \end{aligned} \tag{45}$$

Finally, by substituting (42), (44), and (45) into (34), we have

$$\begin{aligned}\Delta_x^{k+1} &\le \Delta_x^k+\left(3+\frac{2\beta^2}{\mu^2}+\frac{\beta}{\mu}\right)\|\nabla g(\lambda^k)\|^2_{BD^k}+\Delta_\lambda^k-\Delta_\lambda^{k+1}\\ &-\nabla_x L(x^k,\lambda^k)^\intercal\left[A(D^k)^{-1}-\left(\beta+\frac{\ell_L}{2}\right)A^2(D^k)^{-2}\right.\\ &\left.-\frac{\beta(3\mu^2+2\beta^2+\beta\mu)}{m^2\mu^2}I\right]\nabla_x L(x^k,\lambda^k). \end{aligned}$$

This concludes the proof of the proposition. □

Finally, we combine the coupled inequalities in Propositions 12 and 13 and provide the linear convergence rate of FedHybrid using the strong-convexity/concavity of $L(\cdot,\lambda)$ and $g(\cdot)$.

For convenience, we define constants $\underline{\alpha}$ and $\underline{\beta}$ as follows,

$$\begin{aligned}\underline{\alpha} &= \min\{\min_{i\in J_1}\{a_i\},\min_{i\in J_2}\{a_i/(\ell_i+\mu)\}\},\\ \underline{\beta} &= \min\{\min_{i\in J_1}\{b_i\},\min_{i\in J_2}\{b_i(m_i+\mu)\}\}. \end{aligned} \tag{46}$$

These constants provide lower bounds to eigenvalues of matrices $A(D^k)^{-1}$ and BD^k, respectively, since

$$\begin{aligned}\underline{\alpha} &= \min\{\min_{i\in J_1}\{a_i\},\min_{i\in J_2}\{a_i/(\ell_i+\mu)\}\}\\ &\le \min_{i\in[n]} a_i\theta_{\min}((D_i^k)^{-1}) = \theta_{\min}(A(D^k)^{-1}),\\ \underline{\beta} &= \min\{\min_{i\in J_1}\{b_i\},\min_{i\in J_2}\{b_i(m_i+\mu)\}\}\\ &\le \min_{i\in[n]} b_i\theta_{\min}(D_i^k) = \theta_{\min}(BD^k). \end{aligned} \tag{47}$$

Now we show the main theorem stating that FedHybrid finds the optimal solution at a linear rate under Assumption 1.

Theorem 14 (Linear Convergence of FedHybrid): For any given $\mu>0$, under Assumption 1, we suppose that the stepsizes

$\{a_i, b_i\}_{i\in[n]} > 0$ satisfy the following conditions,

$$\begin{aligned}
&a_i \le 1/(22\mu/9 + 2\ell),\ b_i \le \min\{\mu/9, \underline{\alpha}m^2/21\},\ \text{for } i \in J_1,\\
&a_i \le (m_i + \mu)/(22\mu/9 + 2\ell),\\
&b_i \le \min\{\mu/9, \underline{\alpha}m^2/21\}/(\ell_i + \mu),\ \text{for } i \in J_2,
\end{aligned} \tag{48}$$

where $\underline{\alpha}$ is defined in (46). Then for all $k = 0, 1, \ldots, K-1$, the iterates generated from FedHybrid in Algorithm 1 satisfy

$$\Delta^{k+1} \le (1 - \rho)\,\Delta^k,$$

where $\rho = \min\{3\underline{\beta}/(13m + 13\mu), m\underline{\alpha}/2\}$ with $\underline{\beta}$ defined in (46) and Δ^k is the merit function defined in (29).

Proof: If dual stepsizes satisfy (48), β defined in (30) satisfies

$$\beta = \max\{\max_{i\in J_1} b_i, \max_{i\in J_2} b_i(\ell_i + \mu)\} \le \min\left\{\frac{\mu}{9}, \frac{\underline{\alpha}m^2}{21}\right\}. \tag{49}$$

Then if primal stepsizes satisfy (48), using $\beta \le \mu/9$ and $\ell_L = \mu + \ell$ defined in Lemma 7, we have

$$\begin{aligned}
\|A(D^k)^{-1}\| &= \max_{i\in[n]} a_i \|(D_i^k)^{-1}\| \\
&\le \max\left\{\max_{i\in J_1} a_i, \max_{i\in J_2} \frac{a_i}{m_i + \mu}\right\} \\
&\le \frac{1}{2(11\mu/9 + \ell)} \le \frac{1}{2(2\beta + \ell_L)}.
\end{aligned} \tag{50}$$

Next, we show some bounds on a constant $\kappa + 12\ell_g\beta$ and a matrix $Q^k = P^k - (6 + 12\beta\ell_g)\beta/m^2 I$ related to κ and P^k defined in Proposition 13, respectively. By straightforward calculations, with $\beta \le \mu/9$ in (49) and $\ell_g = 1/\mu$, we have

$$\kappa + 12\ell_g\beta = 3 + 2\beta^2/\mu^2 + 13\beta/\mu \le 9/2. \tag{51}$$

By the definition of P^k in Proposition 13, matrix Q^k satisfies

$$\begin{aligned}
Q^k &= A(D^k)^{-1} - \left(\beta + \frac{\ell_L}{2}\right) A^2(D^k)^{-2} \\
&\quad - \frac{(\kappa + 12\ell_g\beta + 6)\beta}{m^2} I \\
&\succeq A(D^k)^{-1} - \left(\beta + \frac{\ell_L}{2}\right) A^2(D^k)^{-2} - \frac{21\beta}{2\,m^2} I \\
&\succeq \frac{1}{2} A(D^k)^{-1} - \left(\beta + \frac{\ell_L}{2}\right) A^2(D^k)^{-2} \\
&= \frac{1}{2} A^{\frac{1}{2}}(D^k)^{-\frac{1}{2}}[I - (2\beta + \ell_L)A(D^k)^{-1}]A^{\frac{1}{2}}(D^k)^{-\frac{1}{2}},
\end{aligned} \tag{52}$$

where the first inequality is due to $\beta \le \mu/9$ in (49) and (51) and the last inequality is due to $\beta \le \underline{\alpha}m^2/21$ in (49) and (47). Thus, by (52), the smallest eigenvalue of Q^k satisfies

$$\begin{aligned}
&\theta_{\min}(Q^k) \\
&\ge \frac{1}{2}\theta_{\min}\left(A^{\frac{1}{2}}(D^k)^{-\frac{1}{2}}[I - (2\beta + \ell_L)A(D^k)^{-1}]A^{\frac{1}{2}}(D^k)^{-\frac{1}{2}}\right) \\
&\ge \frac{1}{2}[1 - (2\beta + \ell_L)\|A(D^k)^{-1}\|]\theta_{\min}(A(D^k)^{-1}) \\
&\ge \frac{1}{4}\theta_{\min}(A(D^k)^{-1}) \ge \frac{\underline{\alpha}}{4},
\end{aligned} \tag{53}$$

where the third and the last inequalities are due to (50) and (47), respectively. Now we combine Propositions 12 and 13 to show the result. By multiplying Proposition 12 by 12 and adding Proposition 13, we have

$$\begin{aligned}
13\Delta_\lambda^{k+1} + \Delta_x^{k+1} &\le 13\Delta_\lambda^k - (6 - 12\ell_g\beta - \kappa)\|\nabla g(\lambda^k)\|^2_{BD^k} \\
&\quad + \Delta_x^k - \|\nabla_x L(x^k, \lambda^k)\|^2_{Q^k},
\end{aligned} \tag{54}$$

where $Q^k = P^k - (6 + 12\beta\ell_g)\beta/m^2 I$. By the m_g-strong concavity of $g(\lambda)$ in Lemma 9 with $m_g = 1/(\mu + \ell)$, we have

$$\|\nabla g(\lambda^k)\|^2 \ge 2m_g(g(\lambda^*) - g(\lambda^k)) = \frac{2}{\mu + \ell}\Delta_\lambda^k. \tag{55}$$

Thus, by (51), we have

$$\begin{aligned}
(6 - 12\ell_g\beta - \kappa)\|\nabla g(\lambda^k)\|^2_{BD^k} &\ge \frac{3}{2}\|\nabla g(\lambda^k)\|^2_{BD^k} \\
&\ge \frac{3}{2}\theta_{\min}(BD^k)\|\nabla g(\lambda^k)\|^2 \ge \frac{3\underline{\beta}}{(\mu + \ell)}\Delta_\lambda^k,
\end{aligned} \tag{56}$$

where the last inequality follows from (47) and (55). Similarly, by the m-strong convexity of $L(\cdot, \lambda^k)$ in Lemma 7, we have

$$\begin{aligned}
\|\nabla_x L(x^k, \lambda^k)\|^2 &\ge 2m(L(x^k, \lambda^k) - L(x^*(\lambda^k), \lambda^k)) \\
&= 2m\Delta_x^k.
\end{aligned} \tag{57}$$

Thus, by the lower bound of Q^k in (53), we have

$$\begin{aligned}
\|\nabla_x L(x^k, \lambda^k)\|^2_{Q^k} &\ge \theta_{\min}(Q^k)\|\nabla_x L(x^k, \lambda^k)\|^2 \\
&\ge \frac{\underline{\alpha}}{4}\|\nabla_x L(x^k, \lambda^k)\|^2 \ge \frac{\underline{\alpha}}{2} m\Delta_x^k,
\end{aligned} \tag{58}$$

where the last inequality follows from (57). Finally, substituting (56) and (58) into (54), we have

$$\begin{aligned}
13\Delta_\lambda^{k+1} + \Delta_x^{k+1} &\le 13\Delta_\lambda^k - \frac{3\underline{\beta}}{\mu + \ell}\Delta_\lambda^k + \left(1 - \frac{m\underline{\alpha}}{2}\right)\Delta_x^k \\
&\le 13\left(1 - \frac{3\underline{\beta}}{13(\mu + \ell)}\right)\Delta_\lambda^k + \left(1 - \frac{m\underline{\alpha}}{2}\right)\Delta_x^k \\
&\le (1 - \rho)(13\Delta_\lambda^k + \Delta_x^k),
\end{aligned}$$

where $\rho = \min\{3\underline{\beta}/(13(\mu + \ell)), m\underline{\alpha}/2\}$. □

Theorem 14 provides a linear (Q-linear) convergence rate with respect to the merit function Δ^k, which consists of the dual optimality gap Δ_λ^k and the primal tracking error Δ_x^k. It ensures that FedHybrid converges linearly to the optimal point, regardless of clients' choices of gradient-type or Newton-type updates. Moreover, the linear rate of FedHybrid holds for any local functions satisfying Assumption 1, which do not need to be i.i.d. among clients. In FedHybrid, clients can choose personalized stepsizes related to properties of local functions under condition (48), which provides more flexibility.

Other distributed methods like ESOM [24] that adopt Newton or quasi-Newton information also have provable linear rates as FedHybrid. The existing superlinearly converging method either works only in centralized settings [47], [48] or requires an expensive inner loop at each iteration in distributed settings [49], [50]. To avoid the computation and communication overhead introduced by an inner loop, in FedHybrid, we choose to adopt a distributed approximation of the true Newton's step and thus

a linear rate is the best we can expect for FedHybrid, even when all agents are Newton-type. The current linear rate coefficient $1-\rho$ is a conservative worst-case analysis, relying on the worst condition number among clients. Let $\iota = \ell/m$ denote the condition number of the global objective $\sum_{i\in[n]} f_i$. It is easy to check that ρ is of order ι^{-3}.

V. Numerical Experiments

In this section, we present experimental results for FedHybrid on convex distributed optimization problems. Specifically, we study least squares and binary classification problems in server-client networks, over synthetic and real-life datasets.

Experimental Setup: We evaluate all methods on three setups with non-i.i.d. data partitioning, where in each setup, there are n clients with d-dimensional decision variables and a total amount $N = \sum_{i\in[n]} N_i$ of data in the network with N_i the size of each local datasets. Now we introduce the setup details.

We begin with a regularized linear regression problem,

$$\min_{\omega\in\mathbb{R}^d} \frac{1}{2N}\sum_{i=1}^{n} \|A_i\omega - y_i\|^2 + \frac{\rho}{2}\|\omega\|^2,$$

where $A_i \in \mathbb{R}^{N_i\times d}$ and $y_i \in \mathbb{R}^{N_i}$ are known as the feature matrix and the response vector at client i, respectively, and $\rho \geq 0$ is the penalty parameter. We introduce a setup with non-i.i.d. data distribution based on the above model.

(1) Linear regression on a synthetic dataset. We set $n = 10$, $d = 3$, and $N_i \sim \text{lognormal}(4, 2) + 50$. At each client i, we generate a scaling value $\eta_i \sim \text{lognormal}(2, 4)$ and set $A_i = \eta_i \widehat{A}_i \Upsilon_i$, where $\widehat{A}_i \in \mathbb{R}^{|\mathcal{D}_i|\times d}$ has elements from $U(0, 1]$ and $\Upsilon_i \in \mathbb{R}^{d\times d}$ is a diagonal scaling matrix. For 6 of the clients, we set $\Upsilon_i = I_d$, while for each of the other 4 clients, we randomly select a $j \in [d]$ to set $[\Upsilon_i]_{jj} = 100$ and set $[\Upsilon_i]_{jj} = 0.01$ for the other two diagonal elements. In this way, the 4 clients have local functions with larger condition numbers. The response $y_i \in \mathbb{R}^{N_i}$ is generated as $y_i = A_i\omega_0 + v_i$, where we generate a fixed $\omega_0 \sim \mathcal{N}(0, I_d)$ and noise $v_i \sim \mathcal{N}(0, I_{N_i})$.

The other two experiments solve binary classification problems using the regularized logistic regression model,

$$\min_{\omega\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{n} [-y_i^\intercal \log h_i - (1-y_i)^\intercal \log(1-h_i)] + \frac{\rho}{2}\|\omega\|^2,$$

where $h_i = 1/(1+\exp(-A_i\omega))$, $A_i \in \mathbb{R}^{N_i\times d}$ and $y_i \in \mathbb{R}^{N_i}$ are the known feature matrix and label vector at client i, respectively, and ρ is the penalty parameter.

(2) Logistic regression on a synthetic dataset. The dataset non i.i.d. Synthetic $(0.5, 0.5)$ is originally introduced in [27]. Here we take $n = 30$ and $d = 15$. For another setup (2'), we set $n = 10$ and $d = 12$.

(3) Logistic regression on a mushroom dataset from UCI. For data preprocessing, we encode categorical features and add a vector of all ones. In this way, we have $d = 99$. We set $n = 8$ and sample relative local dataset sizes from $U(0, 1]$. We set 4 and 3 clients to only have data points with labels 0 and 1, respectively, and 1 client to have data with both labels. We refer to these experiments as setups (1)–(3).

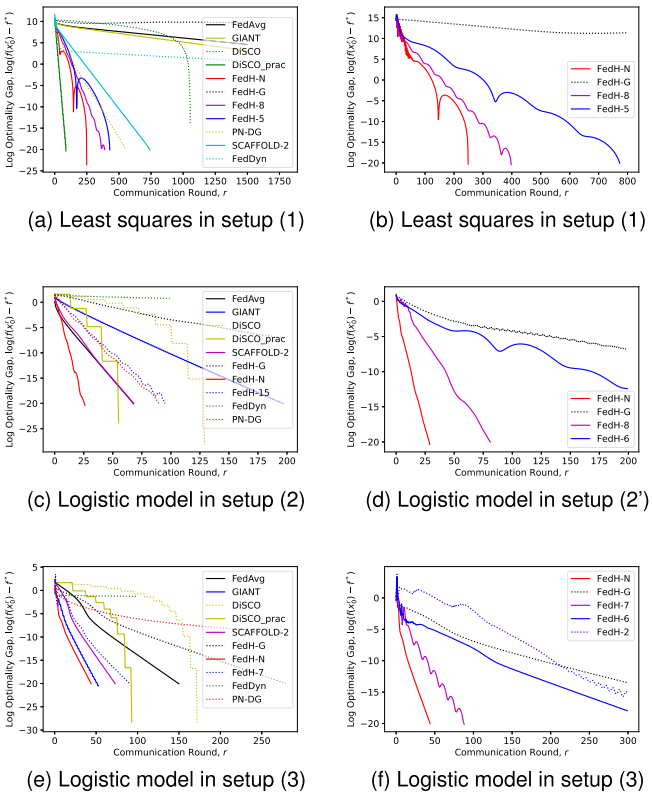
Compared Methods: We conduct experiments on FedHybrid as well as other distributed methods designed for the server-client network. We denote by FedH-K FedHybrid with the number of K clients performing Newton-type updates while all others doing gradient-type updates. We remark that for FedH-K, we have $|J_2| = K$. In particular, FedH-G and FedH-N are short for FedHybrid with all clients performing gradient-type and Newton-type updates, respectively.

We first consider primal iterative methods including FedAvg [5], GIANT [15], and DiSCO [14]. FedAvg is the baseline of first-order methods for the server-client network. We implement the non-stochastic FedAvg, where all clients in the network take a full gradient descent step at each iteration. As a second-order method, GIANT gives the best performance when the clients have i.i.d. local data due to its approximation of the global Newton's direction using the harmonic mean of local Hessians. DiSCO [14] is an inexact Newton's method with damped stepsizes, which uses conjugate gradients as an inner loop to approximate the global Hessian. While the original DiSCO paper proposes to use conservative stepsizes with guaranteed convergence, in practice often a larger stepsize can lead to better performance. Here we implement a practical version named DiSCO_prac, which uses a larger fixed stepsize.

As for the primal-dual methods, we remark that FedPD [32] with all clients taking a full gradient step at each iteration coincides with FedH-G, a special case of FedHybrid when all clients choose gradient-type updates. Other than FedHybrid, we also implement a primal-dual method labeled as PN-DG in figures, inspired by ESOM [24]. In PN-DG, all clients take a primal Newton's step as in (21) and a dual gradient step as in (11) followed by an averaging step by the server at each iteration. Compared with PN-DG, we can show the advantage of the dual Hessian approximation in FedHybrid. We also implement other federated methods, including SCAFFOLD [51] and FedDyn [52], which adopt multiple local gradient updates or argmin of dynamic regularizers, as a comparison of our approximated second-order updates. We remark that since the server and clients only exchange vectors in all these methods, the costs of each communication round for FedHybrid and other methods are the same.

For each setup and each method, we tune parameters and stepsizes using grid search in the range $[10^{-4}, 10]$ and $[10^{-4}, 1]$, respectively, and choose the best one that minimizes the optimality gap. We remark that to save the parameter tuning time, in FedHybrid we use the same stepsizes for clients with the same type of updates, that is, we set $a_i = a_1, b_i = b_1$ for $i \in J_1$ and $a_i = 1, b_i = b_2$ for $i \in J_2$ and only tune stepsizes a_1, b_1, b_2. Here we set $a_i = 1$ for $i \in J_2$ for clients doing Newton-type updates to approximate Newton's method. The optimal value f^* in figures are obtained by the solver and the stopping criteria is set to be $\log(f(x_0^r) - f^*) < -20$.

Observations: In Fig. 1, the x-axis represents the number of communication rounds and the y-axis is the logarithm of the optimality gap. As shown in Fig. 1, with some clients in the network performing Newton-type updates, FedHybrid improves the overall training speed a lot and outperforms the baseline method FedAvg in many scenarios. In particular, if all clients in the network perform Newton-type



(a) Least squares in setup (1) (b) Least squares in setup (1)

(c) Logistic model in setup (2) (d) Logistic model in setup (2')

(e) Logistic model in setup (3) (f) Logistic model in setup (3)

Fig. 1. Performance of FedAvg, DiSCO, SCAFFOLD, FedDyn, & FedH-K.

updates, the second-order FedH-N method achieves a comparable convergence performance as DiSCO_prac. Moreover, FedH-N outperforms PN-DG consistently, which benefits from the dual Hessian approximation utilizing the server-client topology in FedHybrid. The primal method GIANT does not always perform well since its global Hessian approximation can be far from the true Hessian with non-i.i.d. clients. FedH-N also outperforms SCAFFOLD-2 (with two local updates per iteration) and FedDyn consistently, which only adopt first-order information in the updates or the regularizer.

Fig. 1 shows clearly that FedHybrid has a linear performance, which validates the linear rate shown in Theorem 14. Moreover, as shown in the plots on the right column, as the number of Newton-type clients increases, the empirical convergence of FedHybrid is likely to become faster since the local information get better used. This observation suggests that in practical systems, those clients with higher computational capabilities and/or cheaper costs to perform computation can choose to implement Newton-type updates locally to help speed up the overall training speed of the whole system.

In traditional distributed optimization algorithms, all clients perform the same updates. The complexity of the method is determined by the clients equipped with the worst computation hardware. While in the FedHybrid framework, since for a part of the clients, efficient Newton-type updates are involved, the overall system enjoys a faster convergence speed compared to systems running gradient-type methods only. In this sense, FedHybrid utilizes local computation capabilities to improve overall performance. In addition, it provides a novel way to use Newton-type updates rather than multiple local gradient steps to save communication rounds effectively. Thus we can maximally leverage the parallel heterogeneous computation capabilities in this setting.

VI. Final Remarks and Future Works

This paper proposes FedHybrid, a distributed hybrid primal-dual method that allows clients to perform either gradient-type or Newton-type updates based on their computation capacities. We develop a novel merit function for analysis and show a linear convergence rate of FedHybrid for strongly convex objective functions. Numerical studies are also provided to demonstrate the efficacy of FedHybrid in practice.

We highlight a few interesting directions for future work on FedHybrid and distributed optimization. First, an improved linear rate coefficient showing a tradeoff between the number of Newton-type agents and the number of communication rounds might be possible for future works. Second, we could consider stochastic FedHybrid methods. For instance, each client in the network could perform stochastic gradient-type or subsampled Newton-type methods on its local dataset. Moreover, in practice, it is possible that only a small subset of the clients are active at each communication round. Thus, we could consider asynchronous updates in FedHybrid, where only a randomly selected subset of the clients perform updates at each iteration. Also, we expect FedHybrid could be generalized to broader settings, such as time-varying graphs and/or systems with non-convex objective functions.

Appendix

We now show the proofs that are omitted from the paper.

Proof of Lemma 5: By the definition that $L(x,\lambda) = \widetilde{L}(X_0(x,\lambda),x,\lambda)$ and the chain rule in calculus, we have

$$\begin{aligned}\nabla_x L(x,\lambda) &= \nabla_x \widetilde{L}(X_0(x,\lambda),x,\lambda)\\ &= \nabla_x X_0(x,\lambda)\nabla_{x_0}\widetilde{L}(x_0,x,\lambda)\,|_{\,x_0=X_0(x,\lambda)}\\ &\quad + \nabla_x \widetilde{L}(x_0,x,\lambda)\,|_{\,x_0=X_0(x,\lambda)}.\end{aligned}$$

Then we take partial gradient with respect to x_0 in $\widetilde{L}$,

$$\begin{aligned}&\nabla_{x_0}\widetilde{L}(x_0,x,\lambda)\,|_{\,x_0=X_0(x,\lambda)}\\ &\quad= (\mathbb{1}_n\otimes I_d)^\intercal[\lambda+\mu(\mathbb{1}_n\otimes x_0 - x)]\,|_{\,x_0=X_0(x,\lambda)}\\ &\quad= \mu n[x_0 - X_0(x,\lambda)]\,|_{\,x_0=X_0(x,\lambda)} = 0,\end{aligned}$$

where the second equality follows from the definition of $X_0(x,\lambda)$. By combining the above two equations, we have

$$\nabla_x L(x,\lambda) = \nabla_x \widetilde{L}(x_0,x,\lambda)\,|_{\,x_0=X_0(x,\lambda)}.$$

This concludes the proof of the Lemma. □

Proof of Lemma 6: By the first-order optimality condition of the inner problem in Problem 6, we have

$$\begin{aligned} 0 &= \nabla_{\widetilde{x}} \widetilde{L}(\widetilde{x}, \lambda) \,|\, _{\widetilde{x}=\widetilde{x}^*(\lambda)} \\ &= \nabla \widetilde{f}(\widetilde{x}^*(\lambda)) + W^\intercal \lambda + \mu W^\intercal W \widetilde{x}^*(\lambda). \end{aligned}$$

If we consider the first block corresponding to the central decision variable x_0 in the above equation, we have

$$(\mathbb{1}_n^\intercal \otimes I_d)\lambda + \mu n \widetilde{x}_0^*(\lambda) - \mu(\mathbb{1}_n^\intercal \otimes I_d) x^*(\lambda) = 0.$$

Rearranging terms in the above equation, we have

$$\widetilde{x}_0^*(\lambda) = (\mathbb{1}_n^\intercal \otimes I_d)[x^*(\lambda)/n - \lambda/(\mu n)] = X_0(x^*(\lambda), \lambda),$$

where the last equality is the definition of X_0 in (24).

Proof of Lemma 10: Using (27), Lemma 6, and (25), we have

$$\|W\widetilde{x}^k - W\widetilde{x}^*(\lambda^k)\| = \|M(x^k - x^*(\lambda^k))\| \le \|x^k - x^*(\lambda^k)\|,$$

where the inequality is due to (31) and $\|M\| = 1$. Now we upper bound the RHS in the above equation by $\|\nabla_x L(x^k, \lambda^k)\|$. By the m-strong convexity of $L(\cdot, \lambda)$ in Lemma 7, we have

$$\begin{aligned} \|\nabla_x L(x^k, \lambda^k)\| &= \|\nabla_x L(x^k, \lambda^k) - \nabla_x L(x^*(\lambda^k), \lambda^k)\| \\ &\ge m\|x^k - x^*(\lambda^k)\|. \end{aligned}$$

Thus, by combing the preceding two equations, we have

$$\|W\widetilde{x}^k - W\widetilde{x}^*(\lambda^k)\| \le \frac{1}{m}\|\nabla_x L(x^k, \lambda^k)\|.$$

This concludes the proof of the lemma. □

Proof of Lemma 11: Since $2a^\intercal b \le \|a\|^2 + \|b\|^2$ for any a, b,

$$\begin{aligned} &\|W\widetilde{x}^k\|_{BD^k}^2 \\ &\le 2\|W\widetilde{x}^k - W\widetilde{x}^*(\lambda^k)\|_{BD^k}^2 + 2\|W\widetilde{x}^*(\lambda^k)\|_{BD^k}^2 \\ &= 2\|W\widetilde{x}^k - W\widetilde{x}^*(\lambda^k)\|_{BD^k}^2 + 2\|\nabla g(\lambda^k)\|_{BD^k}^2 \\ &\le \frac{2\beta}{m^2}\|\nabla_x L(x^k, \lambda^k)\|^2 + 2\|\nabla g(\lambda^k)\|_{BD^k}^2, \end{aligned} \tag{59}$$

where the equality is due to Lemma 2 and the last inequality is due to (31) and Lemma 10. By taking dual updates in (27),

$$\begin{aligned} \|\lambda^{k+1} - \lambda^k\|^2 &= \|BD^k W\widetilde{x}^k\|^2 \le \beta\|W\widetilde{x}^k\|_{BD^k}^2 \\ &\le \frac{2\beta^2}{m^2}\|\nabla_x L(x^k, \lambda^k)\|^2 + 2\beta\|\nabla g(\lambda^k)\|_{BD^k}^2, \end{aligned}$$

where the inequalities are due to (31) and (59). □

REFERENCES

[1] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*.

[2] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Inform.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[3] K. Zhou and S. I. Roumeliotis, "Multirobot active target tracking with combinations of relative observations," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 678–695, Aug. 2011.

[4] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 592–606, Mar. 2012.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A.Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[6] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends® Mach. Learn.*, Now Publishers, Inc., vol. 14, no. 1-2, pp. 1–210, 2021.

[7] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, vol. 23. Hoboken, NJ, USA: Prentice Hall Englewood Cliffs, 1989.

[8] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

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

[10] T. Chen et al., "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*.

[11] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015.

[12] D. Jakovetić, J. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1131–1146, May 2014.

[13] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1000–1008.

[14] Y. Zhang and X. Lin, "Disco: Distributed optimization for self-concordant empirical loss," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 362–370.

[15] S. Wang, F. Roosta, P. Xu, and M. W. Mahoney, "GIANT: Globally improved approximate newton method for distributed optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 2332–2342.

[16] R. Crane and F. Roosta, "DINGO: Distributed newton-type method for gradient-norm optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[17] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM J. Optim.*, vol. 27, no. 4, pp. 2597–2633, 2017.

[18] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network newton," in *Proc. IEEE 48th Asilomar Conf. Signals, Syst. Comput.*, 2014, pp. 1621–1625.

[19] R. Tutunov, H. Bou-Ammar, and A. Jadbabaie, "Distributed newton method for large-scale consensus optimization," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 3983–3994, Oct. 2019.

[20] C. Sun, M. Ye, and G. Hu, "Distributed optimization for two types of heterogeneous multiagent systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 1314–1324, Mar. 2021.

[21] P. D. Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 2, pp. 120–136, Jun. 2016.

[22] S. Boyd, N. Parikh, and E. Chu, *Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers*. Delft, The Netherlands: Now Publishers Inc, 2011.

[23] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, no. 1, pp. 29–63, 2019.

[24] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 507–522, Dec. 2016.

[25] V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi, "COCOA: A general framework for communication-efficient distributed optimization," *J. Mach. Learn. Res.*, vol. 18, 2018, Art. no. 230.

[26] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-IID data," *Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJxNAnVtDS

[27] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, 2020, pp. 429–450.

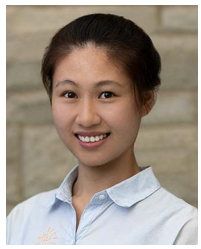
[28] R. Pathak and M. J. Wainwright, "FedSplit: An algorithmic framework for fast federated optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp.7057–7066.

[29] H. Yuan and T. Ma, "Federated accelerated stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp.5332–5344.

[30] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7611–7623.

[31] S. J. Reddi et al., "Adaptive federated optimization," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=LkFG3lB13U5

[32] X. Zhang, M. Hong, S. Dhople, W. Yin, and Y. Liu, "FedPD: A federated learning framework with adaptivity to non-iid data," *IEEE Trans. Signal Process.*, vol. 69, pp. 6055–6070, 2021.
[33] S. Soori, K. Mishchenko, A. Mokhtari, M. M. Dehnavi, and M. Gurbuzbalaban, "DAve-QN: A distributed averaged quasi-newton method with local superlinear convergence rate," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 1965–1976.
[34] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Cambridge, MA, USA: Academic Press, 2014.
[35] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[36] R. A. Tapia, "Diagonalized multiplier methods and quasi-newton methods for constrained optimization," *J. Optim. Theory Appl.*, vol. 22, no. 2, pp. 135–194, 1977.
[37] K. Arrow and L. Hurwicz, *H. Uzawa–Studies in Nonlinear Programming*. Whitefish, MT, USA: Literary Licensing, LLC, 1958.
[38] F. Mansoori and E. Wei, "FlexPD: A flexible framework of first-order primal-dual algorithms for distributed optimization," *IEEE Trans. Signal Process.*, vol. 69, pp. 3500–3512, 2021.
[39] A. S. Berahas, R. Bollapragada, and J. Nocedal, "An investigation of Newton-sketch and subsampled newton methods," *Optim. Methods Softw.*, vol. 35, no. 4, pp. 661–680, 2020.
[40] S. Gong, J. Shen, and L. Du, "Constrained optimization and distributed computation based car following control of a connected and autonomous vehicle platoon," *Transp. Res. Part B: Methodological*, vol. 94, pp. 314–334, 2016.
[41] J. Dongarra, T. Sterling, H. Simon, and E. Strohmaier, "High-performance computing: Clusters, constellations, MPPS, and future directions," *Comput. Sci. Eng.*, vol. 7, no. 2, pp. 51–59, Mar.-Apr. 2005.
[42] Y. Nesterov, "Smooth minimization of non-smooth functions," *Math. Program.*, vol. 103, no. 1, pp. 127–152, 2005.
[43] A. Beck and M. Teboulle, "A fast dual proximal gradient algorithm for convex minimization and applications," *Operations Res. Lett.*, vol. 42, no. 1, pp. 1–6, 2014.
[44] C. A. Uribe, S. Lee, A. Gasnikov, and A. Nedić, "A dual approach for optimal algorithms in distributed optimization over networks," in *Proc. IEEE Inf. Theory Appl. Workshop*, 2020, pp. 1–37.
[45] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized linearized alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 63, no. 15, pp. 4051–4064, Aug. 2015.
[46] S. S. Du and W. Hu, "Linear convergence of the primal-dual gradient method for convex-concave saddle point problems without strong convexity," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 196–205.
[47] G. Zhang, K. Wu, P. Poupart, and Y. Yu, "Newton-type methods for minimax optimization," 2020, *arXiv:2006.14592*.
[48] R. H. Byrd, "Local convergence of the diagonalized method of multipliers," *J. Optim. Theory Appl.*, vol. 26, no. 4, pp. 485–500, 1978.
[49] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network newton distributed optimization methods," *IEEE Trans. Signal Process.*, vol. 65, no. 1, pp. 146–161, Jan. 2017.
[50] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed newton method for network utility maximization–I: Algorithm," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2162–2175, Sep. 2013.
[51] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.
[52] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=B7v4QMR6Z9w



Xiaochun Niu received the B.S. degree from the Department of Mathematics, Nanjing University, Nanjing, China, in 2019. She is currently working toward the Ph.D. degree with the Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA. Her primary research interests include machine learning and optimization and their applications with multiagent systems, with a current emphasis on federated learning, distributed optimization, and minimax problems.

Ermin Wei Wei received the undergraduation triple degree in computer engineering, finance, and mathematics with a minor in German, from the University of Maryland, College Park, MD, USA, the M.S. degree from MIT, and the Ph.D. degree in electrical engineering and computer science with MIT in 2014, advised by Professor Asu Ozdaglar. She is currently an Assistant Professor with the Electrical and Computer Engineering Department and Industrial Engineering and Management Sciences Department, Northwestern University, Evanston, IL, USA. Her research interests include distributed optimization methods, convex optimization and analysis, smart grid, communication systems and energy networks, and market economic analysis. Her team won the 2nd place in the GO-competition Challenge 1, an electricity grid optimization competition organized by Department of Energy.