DISH: A Distributed Hybrid Optimization Method Leveraging System Heterogeneity

Xiaochun Niu and Ermin Wei

***Abstract*—We study distributed optimization problems over multi-agent networks, including consensus and network flow problems. Existing distributed methods neglect the heterogeneity among agents' computational capabilities, limiting their effectiveness. To address this, we propose DISH, a distributed hybrid method that leverages system heterogeneity. DISH allows agents with higher computational capabilities or lower computational costs to perform local Newton-type updates while others adopt simpler gradient-type updates. Notably, DISH covers existing methods like EXTRA, DIGing, and ESOM-0 as special cases. To analyze DISH's performance with general update directions, we formulate distributed problems as minimax problems and introduce GRAND (gradient-related ascent and descent) and its alternating version, Alt-GRAND, for solving these problems. GRAND generalizes DISH to centralized minimax settings, accommodating various descent ascent update directions, including gradient-type, Newton-type, scaled gradient, and other general directions, within acute angles to the partial gradients. Theoretical analysis establishes global sublinear and linear convergence rates for GRAND and Alt-GRAND in strongly-convex-nonconcave and strongly-convex-PL settings, providing linear rates for DISH. In addition, we derive the local superlinear convergence of Newton-based variations of GRAND in centralized settings to show the potentials and limitations of Newton's method in distributed settings. Numerical experiments validate the effectiveness of our methods.**

***Index Terms*—Distributed optimization, heterogeneous systems, hybrid methods, Newton-type methods.**

Manuscript received 1 August 2023; revised 3 February 2024 and 29 May 2024; accepted 15 August 2024. Date of publication 26 August 2024; date of current version 10 September 2024. This work was supported in part by the National Science Foundation under Grant ECCS-2030251 and Grant CMMI-2024774. An earlier version of this paper was presented at the 61st IEEE Conference on Decision and Control [DOI: 10.1109/CDC51059.2022.9993156] and the 58th Annual Allerton Conference on Communication, Control, and Computing [DOI: 10.1109/Allerton49937.2022.9929389]. The associate editor coordinating the review of this article and approving it for publication was Dr. Mingyi Hong. *(Corresponding author: Xiaochun Niu.)*

Xiaochun Niu is with the Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60201 USA (e-mail: xiaochunniu2024@u.northwestern.edu).

Ermin Wei is with the Department of Electrical and Computer Engineering and the Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60201 USA (e-mail: ermin.wei@northwestern.edu).

Digital Object Identifier 10.1109/TSP.2024.3450351

I. Introduction

We study distributed multi-agent optimization problems with communication constraints [3]. These include scenarios like *distributed consensus problems* [4] and *network flow problems*, driven by applications in power grids, sensor networks, communication networks, and machine learning [5], [6]. Agents in distributed computing are located at network nodes and restricted to local data and neighbor communication due to privacy and communication concerns. Their shared goal is to optimize an objective function collaboratively through distributed procedures.

There is a growing literature on developing distributed algorithms, such as gradient-type [7], [8], [9] and Newton-type methods [10], [11], [12], [13] for consensus problems, and methods [14] for network flow problems. However, a notable limitation of existing methods is that they often require all agents to take the same type of updates, leading to bottlenecks caused by agents equipped with slower hardware. This limitation restricts the applicability of fast-converging methods that rely on higher-order computations if even one agent in the system cannot handle them. Nonetheless, heterogeneous configurations are common in modern systems, where advanced processors coexist with older-generation ones, resulting in agents with varying computation capabilities due to hardware constraints. Such heterogeneity presents significant challenges in practical distributed computing systems [15]. Thus, the question arises:

Can we design flexible and efficient distributed hybrid methods to utilize agents' heterogeneous computation capabilities?

We answer the above question affirmatively by proposing DISH, distributed hybrid methods for consensus and network flow problems. DISH utilizes system heterogeneity by allowing agents to choose gradient-type or Newton-type updates based on their computation capabilities. There can be both gradient-type and Newton-type agents in the same communication round, and agents can switch between update types, adapting to their current situation. In particular, when all agents consistently perform Newton-type updates, our hybrid methods for the two problems (consensus and network flow) provide two different ways to approximate the centralized Newton-type descent ascent method (NDA). This opens up opportunities for designing similar hybrid methods tailored to other distributed computing problems. For practical illustration, in cases where the run time for a Newton step, $O(d^3)$ (standard matrix inversion algorithm), at some faster agents, could match the time for a gradient step, $O(d)$, at slower agents, it is reasonable to encourage faster agents to take local second-order updates and speed up the convergence of the entire system. For consensus problems, DISH covers well-known primal-dual gradient-type methods such as EXTRA [7], DIGing [8], and [9], and primal-Newton-dual-gradient methods like ESOM-0 [16] as special cases. It can also be applied to the dual problem of feature-partitioned distributed problems with efficient computation of the conjugate functions. Numerical experiments validate the effectiveness of our hybrid methods, showing faster convergence speeds as the number of Newton-type agents increases.

To analyze the performance of DISH with general update directions, we consider distributed applications as minimax problems and analyze the general gradient-related ascent and descent algorithmic framework (GRAND) for solving minimax problems. GRAND represents the generalization of DISH to centralized minimax settings. We introduce the minimax optimization problem with $L : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}$ strongly convex in x but possibly nonconcave in y:

$$\max_{y\in\mathbb{R}^p} \min_{x\in\mathbb{R}^d} L(x, y). \tag{I.1}$$

The aforementioned distributed optimization problems can be formulated as a form of Problem I.1. Problem I.1 has implications beyond distributed multi-agent optimization and is extensively studied in fields like supervised learning and adversarial training [17]. The *gradient descent ascent method* (GDA) is a simple method for tackling Problem I.1, which performs simultaneous gradient descent on x and gradient ascent on y at each iteration [18], [19]. In addition, Newton-type methods with local superlinear convergence have been proposed [20], [21], [22]. However, existing analyses do not consider a mixture of first and second-order steps. This limitation, along with the demand for distributed hybrid methods, motivates the analysis of GRAND. GRAND allows x and y updates within uniformly bounded acute angles to L's partial gradients. It covers GDA, scaled gradient, Newton-type, and quasi-Newton-type descent ascent methods as special cases. We also introduce the alternating version, Alt-GRAND, where x and y are updated sequentially.

We establish the global sublinear convergence of GRAND and Alt-GRAND for strongly-convex-nonconcave problems. In addition, we demonstrate their linear convergence rates under the assumption of a strongly-convex-Polyak-Łojasiewicz (PL) condition. This condition covers various scenarios, including distributed optimization problems, ensuring the linear rate of DISH. The analysis faces challenges due to the coupled updates of x and y and the time-varying angles between updates and gradients. To tackle these challenges, we bound y's optimality measure and x's tracking error through coupled inequalities. Inspired by two-timescale analysis for bilevel problems, we consider linear combinations of these bounds as Lyapunov functions. Moreover, we examine the local performance of Newton-based methods in centralized settings. In particular, we show the local quadratic rates of the alternating Newton-type method (Alt-NDA) and its variants with multiple x updates. We also present a cubic-rate method that reuses the Hessian inverse for two consecutive steps. These discussions on Newton-based methods aim to clarify the limitations and potentials of distributed second-order methods, which may lead to the future development of superlinear distributed methods.

In summary, to the best of our knowledge, our distributed hybrid methods are the first to allow heterogeneous local updates for distributed consensus and network flow problems with provable convergence and rate guarantees.

A. Related Works

Our work relates to the following growing literature.

Distributed Optimization. For *distributed consensus problems* [4], various first-order iterative methods exist. Distributed (sub)-gradient descent (DGD) [4] combines local gradient descent steps with weighted averaging among neighbors, achieving near-optimal solutions with constant stepsizes. Other methods like EXTRA [7], DIGing [8], and [9] employ gradient tracking techniques and can be viewed as primal-dual gradient methods in augmented Lagrangian formulations, solving exact solutions with constant stepsizes. Second-order primal methods, such as Network Newton [23] and Distributed Newton method [24], approximate Newton steps iteratively through inner loops. Dual decomposition-based methods like ADMM [25], ESOM [16], and PD-QN [26] are also popular. Among them, PD-QN is a primal-dual quasi-Newton method with linear convergence. ESOM is closely related to our DISH method, which combines second-order primal updates with first-order dual updates and demonstrates provable linear convergence. However, none of these methods support heterogeneous agents with different update types. Our earlier work [27] develops a linearly converging distributed hybrid method allowing different update types but relying on the server-client (federated) network structure.

In addition to the consensus problems (sample-partitioned), *feature-partitioned* distributed problems are also prevalent in various fields, including bioinformatics, natural language processing, healthcare, and financial services [25].

Distributed algorithms also tackle *network flow optimization problems* [3] in fields like commodity networks and electric power systems. Existing literature covers first-order methods [14] and second-order methods [28]. Nonetheless, there is a lack of research exploring hybrid methods that enable different update types at network edges or agents.

Minimax Optimization. A simple method for minimax problems is GDA [18]. The monotonicity of the gradient $(\nabla_x L(x, y)^\intercal, -\nabla_y L(x, y)^\intercal)^\intercal$ enables analysis using theorems on monotone operators in variational inequalities [29]. Various first-order methods derived from GDA achieve better performance in different settings, like alternating GDA (Alt-GDA) [30]. Second-order methods exploit Hessian information to accelerate convergence. Some generalize Newton's method from minimization to minimax settings. Studies on Lagrangian problems corresponding to constrained optimization problems demonstrate superlinear local convergence [20], [21].

A complete Newton method with local quadratic rates is proposed [22]. Cubic regularized Newton methods [31] ensure global and local convergence rates by solving minimax subproblems at each iteration. However, global convergence analysis is lacking for Newton-type descent ascent methods without inner loops to solve subproblems or a line search to select stepsizes.

Methods like FR [32] and GDN [22] involve first-order updates on x with second-order updates on y. They converge locally to a minimax point, with GDN showing linear convergence. However, global performance analysis is missing for methods with general update directions.

B. Contributions

As a summary, our contributions are as follows.

1) We propose DISH, a hybrid method for consensus problems (also dual problems of feature-partitioned problems) and network flow problems. DISH leverages agents' heterogeneous computation capabilities by allowing them to choose between gradient-type and Newton-type updates.
2) We establish global sublinear and linear rates for the GRAND frameworks for centralized strongly-convex-nonconcave and strongly-convex-PL minimax problems, ensuring linear convergence for DISH.
3) We examine the local performance of Newton-based methods for centralized minimax problems.
4) Numerical results validate the efficiency of equipping the distributed systems with Newton-type agents.

C. Notation and Outline

For a positive semi-definite matrix A, let $\rho(A)$ be its largest eigenvalue, and $\sigma_{\min}(A)$ ($\sigma_{\min}^{+}(A)$) be its smallest (non-zero) eigenvalue. Let $\mathbb{1}_n$ be the vector of all ones, $\otimes$ be the Kronecker product, and $\circ$ be the function composition. Let $O(\cdot)$ hide constants independent of the target parameter.

The paper is organized as follows. Section II formulates distributed optimization problems as minimax problems and presents the distributed hybrid methods. Section III introduces GRAND and Alt-GRAND for centralized minimax problems. Section IV analyzes the global convergence of GRAND. Section V discusses the local higher-order rates of Newton-based methods. Section VI demonstrates the numerical results.

II. Hybrid Methods for Distributed Optimization

In this section, we introduce distributed optimization problems, including consensus (DC) and network flow (NF) problems. We propose DISH as a distributed hybrid method. In particular, when all agents perform Newton-type updates, the methods for the two problem settings provide two different ways to approximate the Newton-type descent ascent method (NDA) with distributed implementations.

We study optimization problems over multi-agent networks in both DC and NF settings. We define $\mathcal{G}=\{\mathcal{N},\mathcal{E}\}$ as a connected undirected network with the node set $\mathcal{N}=\{1,\dots,n\}$ and the edge set $\mathcal{E}\subseteq\{\{i,j\}\,|\,i,j\in\mathcal{N}, i\neq j\}$. There are n agents in the system, where each agent is located at a node of $\mathcal{G}$ and can only communicate with its neighbors on $\mathcal{G}$ due to privacy issues or communication budgets.

A. Distributed Consensus Problems

This section studies distributed consensus problems. We first formulate the problem in a minimax form.

1) Problem Formulation: In consensus problems, all agents in the network aim to optimize an objective function collaboratively by employing a distributed procedure. Let $\omega\in\mathbb{R}^d$ be the decision variable and $f_i:\mathbb{R}^d\to\mathbb{R}$ be the local function at agent i. We study an optimization problem over $\mathcal{G}$ that $\min_\omega\sum_{i=1}^n f_i(\omega)$. For example, for empirical risk minimization problems in supervised learning, f_i is the empirical loss over local data samples kept at agent i. We impose the following standard assumptions on f_i.

Assumption II.1: The local function f_i is twice differentiable, m_i-strongly convex, and ℓ_i-Lipschitz smooth with constants $0<m_i\le\ell_i<\infty$ for any agent $i\in\mathcal{N}$.

Let $m_{\mathsf{dc}}=\min_{i\in\mathcal{N}}\{m_i\}$ and $\ell_{\mathsf{dc}}=\max_{i\in\mathcal{N}}\{\ell_i\}$. We decouple the computation of individual agent by introducing x_i as the local copy of ω at agent i to develop distributed methods. We formulate *distributed consensus problems* [4] as

$$\min_{x_1,\dots,x_n\in\mathbb{R}^d}\sum_{i=1}^n f_i(x_i)\ \text{ s.t. } x_i=x_j,\ \text{for } \{i,j\}\in\mathcal{E}. \tag{II.1}$$

The consensus constraints $x_i=x_j$ for $\{i,j\}\in\mathcal{E}$ enforce the equivalence of Problem II.1 and the original problem for a connected network $\mathcal{G}$. For compactness, we denote by $x=(x_1^\intercal,\dots,x_n^\intercal)^\intercal$ the concatenation of local variables and $f^{\mathsf{dc}}:\mathbb{R}^{nd}\to\mathbb{R}$ the aggregate function and reformulate Problem II.1 in an equivalent form,

$$\min_{x\in\mathbb{R}^{nd}} f^{\mathsf{dc}}(x)=\sum_{i=1}^n f_i(x_i)\quad\text{s.t. } (Z\otimes I_d)x=x, \tag{II.2}$$

where $Z\in\mathbb{R}^{d\times d}$ is a nonnegative consensus matrix and satisfies the following assumption.

Assumption II.2: Matrix Z corresponding to $\mathcal{G}$ satisfies that
(a) Off-diagonal elements: $z_{ij}\neq 0$ if and only if $\{i,j\}\in\mathcal{E}$;
(b) Diagonal elements: $z_{ii}>0$ for all $i\in\mathcal{N}$;
(c) $z_{ij}=z_{ji}$ for all $i\neq j$ and $i,j\in\mathcal{N}$;
(d) $Z\mathbb{1}_n=\mathbb{1}_n$.

Assumption II.2 is standard for consensus matrices. Let γ be the second largest eigenvalue of Z. By Perron-Frobenius theorem, we have $\rho(Z)=1$, $\gamma<1$, and $\ker(I-Z)=\operatorname{span}\{\mathbb{1}_n\}$. The matrix Z ensures that $(Z\otimes I_d)x=x$ if and only if $x_i=x_j$ for all $\{i,j\}\in\mathcal{E}$ [4]. Let $W=(I_n-Z)\otimes I_d$; thus $\rho(W)<2$, $\sigma_{\min}^{+}(W)=1-\gamma$, and $\ker(W)=\operatorname{span}\{\mathbb{1}_n\otimes y: y\in\mathbb{R}^d\}$. We rewrite the constraint in Problem II.2 as $Wx=0$.

Let variable $y=(y_1^\intercal,\dots,y_n^\intercal)^\intercal$ represent the dual variable with $y_i\in\mathbb{R}^d$ associated with the constraint $z_{ii}x_i-\sum_{j\in\mathcal{N}}z_{ij}x_j=0$ at agent i. We introduce the augmented Lagrangian $L^{\mathsf{dc}}(x,y)$ of Problem II.2 with a constant $\mu\ge 0$,

$$L^{\mathsf{dc}}(x,y)=f^{\mathsf{dc}}(x)+y^\intercal Wx+\mu x^\intercal Wx/2. \tag{II.3}$$

The term $\mu x^\intercal Wx/2$ is a penalty for violating the consensus constraint. The augmented Lagrangian in (II.3) can also be viewed as the Lagrangian associated with a penalized problem $\min_x f^{\mathsf{dc}}(x)+\mu x^\intercal Wx/2$ such that $Wx=0$. It is equivalent to Problem II.2 since $\mu x^\intercal Wx/2$ is zero for any feasible x. By the convexity in Assumption II.1 and Slater's condition, strong duality holds for the penalized problem. Thus, the penalized problem and Problem II.2 are equivalent to the dual problem,

$$\max_{y\in\mathbb{R}^{nd}} \psi^{\mathsf{dc}}(y), \text{ where } \psi^{\mathsf{dc}}(y)=\min_{x\in\mathbb{R}^{nd}} L^{\mathsf{dc}}(x,y), \tag{DC}$$

where we refer to $\psi^{\mathsf{dc}}:\mathbb{R}^{nd}\to\mathbb{R}$ as the dual function and the problem as Problem DC. We now develop distributed methods to solve Problem DC. As we will illustrate after Assumption IV.1, given any $y\in\mathbb{R}^{nd}$, $L^{\mathsf{dc}}(\cdot,y)$ is strongly convex with a unique minimizer. For convenience, for any L in Problem I.1 satisfying such a condition, letting $x^*(y)$ be the unique minimizer for any y, we define $\psi:\mathbb{R}^p\to\mathbb{R}$ as follows,

$$\begin{aligned} x^*(y) &= \operatorname*{argmin}_{x\in\mathbb{R}^d} L(x,y),\\ \psi(y) &= \min_{x\in\mathbb{R}^d} L(x,y)=L(x^*(y),y). \end{aligned} \tag{II.4}$$

The next lemma shows the forms of $\nabla\psi(y)$ and $\nabla^2\psi(y)$, based on the well-known envelope theorem. We show it here for completeness. Let $N:\mathbb{R}^d\times\mathbb{R}^p\to\mathbb{R}^{p\times p}$ be an operator,

$$\begin{aligned} &N(x,y)\\ &\quad=\nabla^2_{yx}L(x,y)[\nabla^2_{xx}L(x,y)]^{-1}\nabla^2_{xy}L(x,y)-\nabla^2_{yy}L(x,y). \end{aligned} \tag{II.5}$$

Lemma II.3: Given any $y\in\mathbb{R}^p$, suppose $L(\cdot,y)$ is strongly convex with a unique minimizer $x^*(y)$. With $x^*(y)$ defined in (II.4) and N defined in (II.5), it holds that $\nabla\psi(y)=\nabla_y L(x^*(y),y)$ and $\nabla^2\psi(y)=-N(x^*(y),y)$.

Lemma II.3 shows that $-N$ can evaluate the Hessian $\nabla^2\psi(y)$ with appropriate arguments. This property allows us to approximate $\nabla^2\psi(y)$ in a distributed manner when designing the hybrid methods.

2) Distributed Hybrid Methods for Consensus Problems: We propose DISH to solve Problem DC. It allows choices of gradient-type and Newton-type updates for each agent at each iteration based on their current computation capabilities. The compact form of DISH shows as follows. At iteration k,

$$\begin{aligned} x^{k+1} &= x^k - AP^k\nabla_x L^{\mathsf{dc}}(x^k,y^k),\\ y^{k+1} &= y^k + BQ^k\nabla_y L^{\mathsf{dc}}(x^k,y^k), \end{aligned} \tag{II.6}$$

where stepsize matrices $A=\operatorname{diag}\{a_1,\ldots,a_n\}\otimes I_d$ and $B=\operatorname{diag}\{b_1,\ldots,b_n\}\otimes I_d$ consist of personalized stepsizes a_i and $b_i>0$ for $i\in\mathcal{N}$ and block diagonal scaling matrices $P^k=\operatorname{diag}\{P_1^k,\ldots,P_n^k\}$ and $Q^k=\operatorname{diag}\{Q_1^k,\ldots,Q_n^k\}$ consist of positive definite local scaling matrices P_i^k and $Q_i^k\in\mathbb{R}^{d\times d}$ for $i\in\mathcal{N}$. Here are some examples of possible scaling matrices:

$$\begin{aligned} &\textit{Primal:}\quad \text{Gradient-type: } P_i^k=I_d;\\ &\qquad\qquad \text{Newton-type: } P_i^k=(\nabla^2 f_i(x_i^k)+\mu I_d)^{-1}.\\ &\textit{Dual:}\quad \text{Gradient-type: } Q_i^k=I_d;\\ &\qquad\qquad \text{Newton-type: } Q_i^k=\nabla^2 f_i(x_i^k)+\mu I_d. \end{aligned} \tag{II.7}$$

Algorithm 1 DISH for Consensus Problems

```
1: Input: Initialization $x_i^0, y_i^0\in\mathbb{R}^d$, stepsizes $a_i, b_i>0$ for $i\in\mathcal{N}$, and $\mu\ge 0$.
2: for $k=0,\ldots,K-1$ do
3:     for each agent $i\in\mathcal{N}$ in parallel do
4:         Send $x_i^k$ and $y_i^k$ to its neighbors $j$ for $\{i,j\}\in\mathcal{E}$;
5:         Choose its local scaling matrices $P_i^k$ and $Q_i^k$;
6:         $x_i^{k+1}=x_i^k-a_iP_i^k[\nabla f_i(x_i^k)+(1-z_{ii})(y_i^k+\mu x_i^k)-\sum_{j:\,\{j,i\}\in\mathcal{E}} z_{ij}(y_j^k+\mu x_j^k)]$;
7:         $y_i^{k+1}=y_i^k+b_iQ_i^k\big[(1-z_{ii})x_i^k-\sum_{j:\,\{i,j\}\in\mathcal{E}} z_{ij}x_j^k\big]$.
8:     end for
9: end for
```

We refer to [1] for a detailed explanation of the choices of local scaling matrices. We define two cases of DISH: DISH-G, where all agents perform gradient-type updates (which is equivalent to GDA), and DISH-N, which approximates the Newton-type descent ascent method (NDA) with a distributed procedure since the primal (for $\mu>0$) and dual Hessians are inseparable. In addition to gradient-type and Newton-type updates, DISH allows agents to take other local updates, such as scaled gradient or quasi-Newton directions. Algorithm 1 presents the distributed implementation of DISH by substituting the partial gradients in (II.6). It includes a primal step (Line 6) and a dual step (Line 7) at each agent. Moreover, an alternating version of DISH under the Alt-GRAND framework is ensured to converge. Another practical variant is when agent i obtains y_i^{k+1} using the updated x_i^{k+1}, some x_j^k, and some updated x_j^{k+1} from its neighbors.

DISH covers existing distributed methods such as EXTRA [7], DIGing [8], [9], and ESOM-0 [16] through appropriate parameter choices. More details on these relationships can be found in [1]. DISH allows agents with higher computational capabilities or cheaper computational costs to locally implement Newton-type updates, while others can adopt simpler gradient-type updates. It provides flexibility by allowing agents to use different types of updates across iterations and between primal and dual spaces within the same iteration. Numerical studies in Section VI-A show that DISH achieves faster performance when more agents adopt Newton-type updates since it better utilizes local information. It is worth noting that Algorithm 1 offers alternative ways to develop distributed methods beyond the choices in (II.7). For instance, PD-QN [26], which matches the linear rate of DISH, approximates the primal-dual quasi-Newton method using distributable matrices that satisfy the quasi-Newton (global secant) conditions. PD-QN is a special case of Algorithm 1 since its scaling matrices are uniformly lower and upper-bounded.

3) Feature-Partitioned Distributed Problems: We consider prediction problems over $\mathcal{G}$ and denote by $\Theta\in\mathbb{R}^{N\times d}$ the input data matrix with N samples and d features. Then Problem II.1 corresponds to sample-partitioned settings with partitioned data $\Theta=(\theta_1^\intercal,\ldots,\theta_n^\intercal)^\intercal$, where a row block $\theta_i\in\mathbb{R}^{N_i\times d}$ represents the N_i local samples kept at agent i and $\sum_{i\in\mathcal{N}} N_i=N$. Alternatively, in feature-partitioned settings [25], the data matrix is split into $\Theta = (\Theta_1, \ldots, \Theta_n)$, where a column block $\Theta_i \in \mathbb{R}^{N\times d_i}$ is the d_i local features kept at agent i and $\sum_{i\in\mathcal{N}} d_i = d$. In this setting, each agent has access to the entire set of data samples but only a unique subset of the features. The previous section presents DISH to solve sample-partitioned consensus problems, and now we consider its extension to feature-partitioned distributed settings.

Feature-partitioned problems are likely to involve a moderate number of samples and a large number of features [25]. For example, scientists can collaboratively study DNA mutations using a few volunteers' DNA data recorded at multiple labs; and doctors may evaluate shared patients' health conditions by leveraging their medical data from several specialists.

Given partitioned data $\Theta = (\Theta_1, \ldots, \Theta_n) \in \mathbb{R}^{N\times d}$ and $\Theta_i \in \mathbb{R}^{N\times d_i}$, we decompose the decision variable as $\xi = (\xi_1^\intercal, \ldots, \xi_n^\intercal)^\intercal \in \mathbb{R}^d$ with $\xi_i \in \mathbb{R}^{d_i}$. This gives $\Theta\xi = \sum_{i\in\mathcal{N}} \Theta_i\xi_i$. We consider a convex loss function ϕ and a convex and separable regularizer r such that $r(\xi) = \sum_{i\in\mathcal{N}} r_i(\xi_i)$. Examples of separable regularizers include the l_2 norm $\|\xi\|^2 = \sum_{i\in\mathcal{N}} \|\xi_i\|^2$. We can formulate the optimization problem of the feature-partitioned scenario as follows,

$$\min_{\xi\in\mathbb{R}^d} \phi\left(\sum_{i=1}^n \Theta_i\xi_i\right) + \sum_{i=1}^n r_i(\xi_i). \tag{II.8}$$

Let $f^*(\lambda) = \max_x\{\lambda^\intercal x - f(x)\}$ be the convex dual conjugate of any function f. The following proposition shows that the dual problem of Problem II.8 takes the form of the consensus problem in (II.2). Similar results are also shown in [25].

Proposition II.4: Problem II.8 is equivalent to the following problem with $x = (x_1^\intercal, \ldots, x_n^\intercal)^\intercal \in \mathbb{R}^{nN}$ and a consensus matrix Z corresponding to graph $\mathcal{G}$,

$$\min_x \sum_{i=1}^n \left[r_i^*(-\Theta_i^\intercal x_i) + \phi^*(x_i)/n\right], \text{s.t. } [(I_n - Z)\otimes I_d]x = 0.$$

Proposition II.4 shows the equivalence between Problem II.8 and a form of Problem II.2, which is equivalent to Problem DC. This suggests that if the gradients (and Hessians) of conjugates ϕ^* and r_i^* can be computed efficiently in practice (e.g., by a closed form or polynomial-time algorithms), we can apply DISH to solve the corresponding dual problem in Proposition II.4 instead of the original one in (II.8). Here is an example of when the conjugates can be easily computed.

Example II.5: Suppose that $\omega \in \mathbb{R}^N$ and $\omega_i \in \mathbb{R}^{d_i}$, and quadratic functions $\phi(\omega) = \omega^\intercal U\omega/2 + u^\intercal\omega$ and $r_i(\omega_i) = \omega_i^\intercal V_i\omega_i/2 + v_i^\intercal\omega_i$ with $U \in \mathbb{R}^{N\times N} \succ 0$ and $V_i \in \mathbb{R}^{d_i\times d_i} \succ 0$ for $i \in \mathcal{N}$. It is easy to compute the conjugates ϕ^* and r_i^* and obtain the dual problem in Proposition II.4 that $\min_x \sum_{i=1}^n [(\Theta_i^\intercal x_i - v_i)^\intercal V_i^{-1}(\Theta_i^\intercal x_i - v_i) + (x_i - u)^\intercal U^{-1}(x_i - u)/n]/2$ such that $[(I_n - Z)\otimes I_d]x = 0$. In DISH, we have $P_i^k = Q_i^k = I_N$ for gradient-type updates, and $(P_i^k)^{-1} = Q_i^k = \Theta_i V_i^{-1}\Theta_i^\intercal + U^{-1}/n + \mu I_N$ for Newton-type updates. Thus, when $r_i(\omega_i)$ is the l_2 regularizer with $V_i = \chi I_{d_i}$ and $\chi > 0$, and the number of samples N is relatively small, we can compute the Newton-type updates efficiently.

B. Network Flow Optimization Problems

We now study nonlinear network flow optimization problems over multi-agent networks. We first present the problem setting and its equivalent structured minimax formulation.

1) Problem Formulation: We recall that agent i locates at node i in the network. In a network flow problem, we define $x \in \mathbb{R}^{|\mathcal{E}|}$ as the decision variable with entries x_{ij} for $\{i,j\} \in \mathcal{E}$. For a convention, we ask agent i to control the flow x_{ij} for any $j > i$, and we use x_{ij} for $i < j$ to denote the directed flow from node i to node j. Let $\pi \in \mathbb{R}^n$ be a given supply vector with entries π_i the external supply (demand) when $\pi_i > 0$ ($\pi_i < 0$) at agent i. We assume $\sum_{i\in\mathcal{N}} \pi_i = 0$ to ensure the total supply equals the total demand over the system. We suppose the cost function is separable in terms of edges with the form $f^{\mathsf{nf}}(x) = \sum_{\{i,j\}\in\mathcal{E}} f_{ij}(x_{ij})$, where $f_{ij}: \mathbb{R} \to \mathbb{R}$ is the cost at edge $\{i,j\}$. We study a separable *network flow optimization problem* [3] with the flow balance constraint,

$$\begin{aligned}&\min_{x\in\mathbb{R}^{|\mathcal{E}|}} \sum_{\{i,j\}\in\mathcal{E}} f_{ij}(x_{ij}),\\ &\text{s.t.} \sum_{j:\,\{i,j\}\in\mathcal{E},j>i} x_{ij} - \sum_{j:\,\{i,j\}\in\mathcal{E},j<i} x_{ij} = \pi_i,\ \forall i \in \mathcal{N}.\end{aligned} \tag{II.9}$$

By summing up the constraints over all $i \in \mathcal{N}$, we verify that $\sum_{i\in\mathcal{N}} \pi_i = 0$ as required before. Let $E \in \mathbb{R}^{n\times|\mathcal{E}|}$ denote the node-edge incidence matrix with entries $E_{i,\{i,j\}} = 1$ and $E_{j,\{i,j\}} = -1$ if $i < j$ and $E_{k,\{i,j\}} = 0$ if $k \neq i, j$ for $\{i,j\} \in \mathcal{E}$. We remark that $\ker(E^\intercal) = \text{span}\{\mathbb{1}_n\}$ and the Laplacian matrix of $\mathcal{G}$ can be represented as $EE^\intercal \in \mathbb{R}^{n\times n}$. For compactness, we rewrite Problem II.9 as follows,

$$\min_{x\in\mathbb{R}^{|\mathcal{E}|}} f^{\mathsf{nf}}(x), \quad \text{s.t. } Ex = \pi. \tag{II.10}$$

Since $\text{im}(E) = \ker(E^\intercal)^\perp = \text{span}\{\mathbb{1}_n\}^\perp$ and $\mathbb{1}_n^\intercal\pi = 0$, we have $\pi \in \text{im}(E)$. Thus, there exists a feasible x^{nf} to the above problem such that $Ex^{\mathsf{nf}} = \pi$. We impose the following assumption on f^{nf}.

Assumption II.6: The function $f^{\mathsf{nf}}(x)$ is twice differentiable, m_{nf}-strongly convex, and ℓ_{nf}-Lipschitz smooth with constants $0 < m_{\mathsf{nf}} \le \ell_{\mathsf{nf}}$.

Let $y = (y_1; \cdots; y_n) \in \mathbb{R}^n$ be the dual variable with y_i associated with the constraint $[Ex]_i = \pi_i$ at agent i. To solve Problem II.10 with the flow balance constraint, we define the Lagrangian L^{nf} as follows,

$$L^{\mathsf{nf}}(x,y) = f^{\mathsf{nf}}(x) + y^\intercal(Ex - \pi). \tag{II.11}$$

By the convexity of f^{nf} and Slater's condition, strong duality holds. Thus, Problem II.10 is equivalent to the following minimax problem, which we refer to as Problem NF,

$$\max_{y\in\mathbb{R}^n} \psi^{\mathsf{nf}}(y), \text{ where } \psi^{\mathsf{nf}}(y) = \min_{x\in\mathbb{R}^{|\mathcal{E}|}} L^{\mathsf{nf}}(x,y). \tag{NF}$$

As will illustrate after Assumption IV.1, given any $y \in \mathbb{R}^p$, $L^{\mathsf{nf}}(\cdot, y)$ is strongly convex with a unique minimizer.

2) Distributed Hybrid Methods for Network Flow: We propose distributed hybrid methods for solving Problem NF. The hybrid method allows various updating types for primal variables at each iteration. By substituting $\nabla_x L^{\mathsf{nf}} = \nabla f^{\mathsf{nf}}(x) + E^\intercal y$ and $\nabla_y L^{\mathsf{nf}} = Ex - \pi$, the compact form of the distributed hybrid method performs as follows,

$$\begin{aligned} x^{k+1} &= x^k - AP^k(\nabla f^{\mathsf{nf}}(x^k) + E^\intercal y^k), \\ y^{k+1} &= y^k + BQ^k(Ex^k - \pi), \end{aligned} \tag{II.12}$$

where $A = \operatorname{diag}\{a_{ij}\} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ and $B = \operatorname{diag}\{b_i\} \in \mathbb{R}^{n\times n}$ consist of positive stepsizes a_{ij} for $\{i,j\}\in\mathcal{E}$ and b_i for $i\in\mathcal{N}$, and $P^k = \operatorname{diag}\{p_{ij}^k\} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ and $Q^k = \operatorname{diag}\{q_i^k\} \in \mathbb{R}^{n\times n}$ consist of positive scaling values p_{ij}^k for $\{i,j\}\in\mathcal{E}$ and q_i^k for $i\in\mathcal{N}$. The scaling values here serve more like personalized stepsizes for each variable and each iteration. Here are examples of possible gradient-type and Newton-type scaling values. We define $\mathcal{J}_1^k = \{\{i,j\}\in\mathcal{E}\colon x_{ij}$ takes gradient-type updates at iteration $k\}$ and $\mathcal{J}_2^k = \{\{i,j\}\in\mathcal{E}\colon x_{ij}$ takes Newton-type updates at $k\}$. We take

$$\begin{aligned} &\textit{Primal:}\quad \text{Gradient-type, } p_{ij}^k = 1; \\ &\qquad\qquad \text{Newton-type, } p_{ij}^k = (\nabla^2 f_{ij}(x_{ij}^k))^{-1},\ \forall\{i,j\}\in\mathcal{E}. \\ &\textit{Dual:}\quad q_i^k = \Big[|\{j\colon \{i,j\}\in\mathcal{J}_1^k\}| \\ &\qquad\qquad + \sum_{j\colon \{i,j\}\in\mathcal{J}_2^k} (\nabla^2 f_{ij}(x_{ij}^k))^{-1}\Big]^{-1},\ \forall i\in\mathcal{N}. \end{aligned} \tag{II.13}$$

We will illustrate the scalings in (II.13) in the next subsection. Other choices of positive scaling values uniformly bounded over k can also work. Algorithm 2 shows the distributed implementation of (II.12). It consists of primal (Lines 6–8) and dual steps (Line 9) for each agent. The primal step on x_{ij} using p_{ij}^k in (II.13) reflects the flow on edge $\{i,j\}\in\mathcal{E}$ and is updated using either gradient-type or Newton-type information of its local edge objective f_{ij} along with the two end points' dual variables. The dual step on y_i using q_i^k given by (II.13) corresponds to the flow balance constraint at node i and uses all the primal information from its neighboring edges x_{ij}.

3) Special Cases of Algorithm 2: We now illustrate the update choices provided in (II.13). We begin with the two extreme cases with all gradient or Newton-type edges. First, when all edges take gradient-type updates, (II.13) implies that $P^k = I_{|\mathcal{E}|}$ and $BQ^k = \operatorname{diag}\{b_i/\deg(i)\}$ for any k in (II.12). It recovers GDA with personalized stepsizes for y_i.

Next, we consider the case when all edges take Newton-type updates for a speedup. We have $q_i^k = \sum_{j\colon \{i,j\}\in\mathcal{E}} (\nabla^2 f_{ij}(x_{ij}^k))^{-1}$ for $i\in\mathcal{N}$ in (II.13) and

$$P^k = (\nabla^2 f^{\mathsf{nf}}(x^k))^{-1} \text{ and } Q^k = \operatorname{diag}\{q_i^k\}. \tag{II.14}$$

We now study both the primal and the dual updates and show that Algorithm 2 approximates NDA by a diagonalized dual Hessian in this particular case.

Primal Updates. The primal Newton's step for solving the inner problem $\min_x L^{\mathsf{nf}}(x,y)$ in (NF) at iteration k is

$$x^{k+1} = x^k - \left(\nabla_{xx}^2 L^{\mathsf{nf}}(x^k,y^k)\right)^{-1}\nabla_x L^{\mathsf{nf}}(x^k,y^k).$$

Algorithm 2 Distributed Hybrid Method for Network Flow Optimization

```
1:  Input: Initialization $x_{ij}^0, y_i^0 \in \mathbb{R}$ and stepsizes $a_{ij}, b_i \in \mathbb{R}^+$
    for $\forall i\in\mathcal{N}$ and $\forall\{i,j\}\in\mathcal{E}$, respectively.
2:  for $k = 0,\ldots,K-1$ do
3:    for each agent $i\in\mathcal{N}$ in parallel do
4:      Send values $x_{ij}^k$ and $y_i^k$ (and $(\nabla^2 f_{ij}(x_{ij}^k))^{-1}$ if
        $\{i,j\}\in\mathcal{J}_2^k$) to $i$'s neighbor $j$;
5:      Choose its local scale values $p_{ij}^k$ and $q_i^k$;
6:      for each neighbor $j$ ($j$ such that $\{i,j\}\in\mathcal{E}$) satisfying
        $j > i$ in parallel do
7:        $x_{ij}^{k+1} = x_{ij}^k - a_{ij}p_{ij}^k(\nabla f_{ij}(x_{ij}^k) + y_i^k - y_j^k)$;
8:      end for
9:      $y_i^{k+1} = y_i^k + b_i q_i^k(\sum_{j\colon \{i,j\}\in\mathcal{E}, j>i} x_{ij}^k -$
        $\sum_{j\colon \{i,j\}\in\mathcal{E}, j<i} x_{ij}^k - \pi_i)$;
10:   end for
11: end for
```

By substituting $\nabla_{xx}^2 L^{\mathsf{nf}}(x^k,y^k) = \nabla^2 f^{\mathsf{nf}}(x^k)$ and $\nabla_x L^{\mathsf{nf}}(x^k,y^k) = \nabla f^{\mathsf{nf}}(x^k) + E^\intercal y^k$ in NDA, it recovers the primal Newton's step with P^k given in (II.14).

Dual Updates. We now consider y's (dual) Newton's update for $\max_y \psi^{\mathsf{nf}}(y)$ at iteration k. We replace $x^*(y^k)$ by the current primal iterate x^k and define $\widehat{\nabla}\psi^{\mathsf{nf}}(y^k)$ and $\widehat{\nabla}^2\psi^{\mathsf{nf}}(y^k)$ as estimators of $\nabla\psi^{\mathsf{nf}}(y^k)$ and $\nabla^2\psi^{\mathsf{nf}}(y^k)$ due to the lack of the exact minimizer $x^*(y^k)$, and obtain

$$\begin{aligned} \widehat{\nabla}\psi^{\mathsf{nf}}(y^k) &= Ex^k - \pi, \\ \widehat{\nabla}^2\psi^{\mathsf{nf}}(y^k) &= -N^{\mathsf{nf}}(x^k,y^k) = -E[\nabla^2 f^{\mathsf{nf}}(x^k)]^{-1}E^\intercal. \end{aligned}$$

We remark that $\widehat{\nabla}^2\psi^{\mathsf{nf}}(y^k)$ is not full-rank due to the matrix E. Let Δy^k be dual Newton's update that $y^{k+1} = y^k + \Delta y^k$ defined by $\widehat{\nabla}^2\psi^{\mathsf{nf}}(y^k)\Delta y^k = -\widehat{\nabla}\psi^{\mathsf{nf}}(y^k)$. Then it satisfies

$$E[\nabla^2 f^{\mathsf{nf}}(x^k)]^{-1}E^\intercal\Delta y^k = Ex^k - \pi.$$

Since the dual Hessian $E[\nabla^2 f^{\mathsf{nf}}(x^k)]^{-1}E^\intercal$ is inseparable, we approximate it by its diagonal part to design a distributed method. We recall that the Laplacian matrix of $\mathcal{G}$ is $EE^\intercal = D - A^{\mathsf{adj}}$, where D is the degree matrix with diagonal entries $D_{ii} = \deg(i)$ for $i\in\mathcal{N}$ and 0 otherwise, and A^{adj} is the adjacency matrix with entries $A_{ij}^{\mathsf{adj}} = 1$ if $\{i,j\}\in\mathcal{E}$ and 0 otherwise. Inspired by this, we split by $E[\nabla^2 f^{\mathsf{nf}}(x^k)]^{-1}E^\intercal = D^{\mathsf{nf}}(x^k) - A^{\mathsf{nf}}(x^k)$, where $D^{\mathsf{nf}}(x^k)$ is diagonal with $[D^{\mathsf{nf}}(x^k)]_{ii} = \sum_{j\colon \{i,j\}\in\mathcal{E}} (\nabla^2 f_{ij}(x_{ij}^k))^{-1}$ for $i\in\mathcal{N}$ and $A^{\mathsf{nf}}(x^k)$ has entries $[A^{\mathsf{nf}}(x^k)]_{ij} = (\nabla^2 f_{ij}(x_{ij}^k))^{-1}$ if $\{i,j\}\in\mathcal{E}$ and 0 otherwise. We approximate $E[\nabla^2 f^{\mathsf{nf}}(x^k)]^{-1}E^\intercal$ by its diagonal part $D^{\mathsf{nf}}(x^k)$ to obtain a distributed scheme. We note that $Q^k = [D^{\mathsf{nf}}(x^k)]^{-1}$ in (II.14). Thus, the dual Newton-type updates with Q^k in (II.14) estimate Newton's steps by adopting the diagonalized Hessian.

We further discuss the updates P^k and Q^k provided in (II.13) when the system has both gradient-type and Newton-type edges. The diagonal matrix P^k denotes whether the local update is gradient-type ($p_{ij}^k = 1$) or Newton-type ($p_{ij}^k = (\nabla^2 f_{ij}(x_{ij}^k))^{-1}$). Moreover, similar to the Newton-type dual

Algorithm 3 GRAND: Gradient-Related Ascent and Descent.

```
1: Input: $\alpha > 0$, $\beta > 0$, $x^0 \in \mathbb{R}^d$, and $y^0 \in \mathbb{R}^p$.
2: for $k = 0, \cdots, K-1$ do
3:     Take $s^k \in \mathbb{R}^d$ and $t^k \in \mathbb{R}^p$ satisfying Assumption III.1
       (or $s^k \in \mathbb{R}^d$ under Assumption III.2 for Alt-GRAND)
4:     $x^{k+1} = x^k - \alpha s^k$,
5:     (Take $t^k \in \mathbb{R}^p$ under Assumption III.2 for Alt-GRAND)
6:     $y^{k+1} = y^k + \beta t^k$.
7: end for
```

updates, $(Q^k)^{-1}$ in (II.13) takes the diagonal part of the matrix $EP^kE^\intercal$ to utilize the primal gradient or Hessian information from adjacent edges as defined in P^k. In summary, Algorithm 2 provides a flexible distributed method when there are both gradient-type and Newton-type edges in the system.

III. GRAND: Gradient-Related Ascent and Descent Algorithm

Recall that Problems DC and NF are in the minimax form of Problem I.1. Thus, to analyze the performance of our distributed hybrid methods with general update directions, we analyze generalized methods for solving Problem I.1.

A. GRAND

We introduce the *gradient-related ascent and descent* (GRAND) algorithmic framework in Algorithm 3 for solving minimax problems. GRAND presents a generalization of the distributed hybrid methods proposed in Algorithms 1 and 2. In Algorithm 3, constants α and β are stepsizes and vectors s^k and t^k are x-descent and y-ascent update directions, respectively. GRAND generalizes the gradient descent ascent method (GDA) by allowing updates s^k and t^k to be within uniformly bounded acute angles to the partial gradients. We state the formal assumptions as follows.

Assumption III.1: There are positive constants γ_s, γ_t, Γ_s, and Γ_t such that for any k, the updates s^k and t^k satisfy

$$\|s^k\| \geq \sqrt{\gamma_s \Gamma_s}\|\nabla_x L(x^k, y^k)\|,$$
$$(s^k)^\intercal \nabla_x L(x^k, y^k) \geq \|s^k\|^2/\Gamma_s,$$
$$\|t^k\| \geq \sqrt{\gamma_t \Gamma_t}\|\nabla_y L(x^k, y^k)\|,$$
$$(t^k)^\intercal \nabla_y L(x^k, y^k) \geq \|t^k\|^2/\Gamma_t.$$

Assumption III.1 is inspired by the gradient-related descent methods for solving minimization problems [3]. For the x-update s^k, the first condition implies that $s^k \neq 0$ and thus $x^{k+1} \neq x^k$ whenever $\nabla_x L(x^k, y^k) \neq 0$, and the second condition ensures that $-s^k$ is a descent direction with an acute angle to $\nabla_x L(x^k, y^k)$. Similarly, t^k is an ascent direction along $\nabla_y L(x^k, y^k)$. We will provide a general convergence analysis of GRAND in Section IV-B. GRAND is a general framework that includes some important specific methods. We first note that GDA is a special case of GRAND.

GDA. If we take $s^k = \nabla_x L(x^k, y^k)$ and $t^k = \nabla_y L(x^k, y^k)$ for all k, Algorithm 3 recovers GDA with $\gamma_s = \Gamma_s = \gamma_t = \Gamma_t = 1$ in Assumption III.1.

Besides the gradient method, the gradient-related directions also enable methods adopting scaled gradients, Newton's updates, or quasi-Newton updates. These methods can potentially improve the local numerical performance.

Scaled Gradient Descent Ascent Method. Algorithm 3 leads to the scaled gradient method when $s^k = P^k \nabla_x L(x^k, y^k)$ and $t^k = Q^k \nabla_y L(x^k, y^k)$ with positive definite scaling matrices P^k and Q^k. We assume uniformly bounded eigenvalues of P^k and Q^k such that $\sqrt{\gamma_s \Gamma_s} I_d \preceq P^k \preceq \Gamma_s I_d$ and $\sqrt{\gamma_t \Gamma_t} I_p \preceq Q^k \preceq \Gamma_t I_p$ for all k to satisfy Assumption III.1.

The scalings P^k and Q^k provide flexibility when designing distributed methods. They can help the system mimic Newton's update and improve numerical performance. In particular, our distributed hybrid methods proposed in Algorithms 1 and 2 are special cases of GRAND with scaled gradient updates.

Moreover, if $P^k = P$ and $Q^k = Q$ are constant matrices, they are also known as the preconditioners. Preconditioners are shown to be crucial in practice when training GANs [32].

Newton-type Descent Ascent Method (NDA). Algorithm 3 is Newton-type when $s^k = [\nabla^2_{xx} L(x^k, y^k)]^{-1} \nabla_x L(x^k, y^k)$ and $t^k = [N(x^k, y^k)]^{-1} \nabla_y L(x^k, y^k)$. Here $N(x^k, y^k)$ estimates the Hessian $-\nabla^2 \psi(y^k)$ by replacing $x^*(y^k)$ with x^k. In this method, x takes a Newton's step along $\nabla_x L(x^k, y^k)$ and moves towards $x^*(y^k)$, and y mimics the Newton's step $-[\nabla^2 \psi(y^k)]^{-1} \nabla \psi(y^k)$ to maximize $\psi(y)$.

Assumption III.1 holds for s^k with $\Gamma_s = 1/m_{\mathsf{x}}$ and $\sqrt{\gamma_s \Gamma_s} = 1/\ell_{\mathsf{xx}}$ under Assumption IV.1. Moreover, it holds for t^k, if there exists a constant $\varrho > 0$ such that $N(x,y) \succ \varrho I_p$ for any (x, y). In this case, we have $\Gamma_t = 1/\varrho$ and $\sqrt{\gamma_t \Gamma_t} = 1/(\ell_{\mathsf{yx}} \ell_{\mathsf{xy}}/m_{\mathsf{x}} + \ell_{\mathsf{yy}})$. Such a condition is not restrictive. For example, when there is $m_{\mathsf{y}} > 0$ such that $L(x,y)$ is m_{y}-strongly concave with respect to y, we have $\nabla^2_{yy} L(x,y) \preceq -m_{\mathsf{y}} I_p$. If we further assume the continuity of $\nabla^2_{yx} L(x,y)$, we have $\nabla^2_{yx} L(x,y) = (\nabla^2_{xy} L(x,y))^\intercal$ by Clairaut's theorem. Thus, we have $\nabla^2_{yx} L(x,y) [\nabla^2_{xx} L(x,y)]^{-1} \nabla^2_{xy} L(x,y) \succeq 0_p$ since $[\nabla^2_{xx} L(x,y)]^{-1} \succeq 1/\ell_{\mathsf{xx}} \cdot I_d$. In this case, we can take $\varrho = m_{\mathsf{y}}/2$ and Assumption III.1 holds for N defined in (II.5).

Quasi-Newton-type Descent Ascent Method. The aforementioned scalings P^k and Q^k can also be quasi-Newton updates, like (L)-BFGS matrices. For example, PD-QN [26], the distributed primal-dual quasi-Newton method for consensus problems is a special case of GRAND with Assumption III.1.

B. Alternating GRAND

We now introduce Alt-GRAND as an alternating version of GRAND, where the updates for x and y are performed sequentially (Gauss-Seidel updates [3]) instead of simultaneously (Jacobi updates). Alt-GRAND adopts the updates in Algorithm 3 with a different assumption that the y-update t^k is along the alternating partial gradient using the updated x^{k+1}. Formally, we present the following assumption.

Assumption III.2 (Alt-GRAND): There are positive constants γ_s, γ_τ, Γ_s and Γ_τ such that s^k and t^k in Algorithm 3 satisfy $\|s^k\| \geq \sqrt{\gamma_s \Gamma_s}\|\nabla_x L(x^k, y^k)\|$, $(s^k)^\intercal \nabla_x L(x^k, y^k) \geq \|s^k\|^2/\Gamma_s$, $\|t^k\| \geq \sqrt{\gamma_\tau \Gamma_\tau}\|\nabla_y L(x^{k+1}, y^k)\|$, and $(t^k)^\intercal \nabla_y L(x^{k+1}, y^k) \geq \|t^k\|^2/\Gamma_\tau$.

We note that in Alt-GRAND, s^k satisfies the same conditions as in GRAND, while t^k is an ascent direction along the updated $\nabla_y L(x^{k+1}, y^k)$ instead of $\nabla_y L(x^k, y^k)$. Alt-GRAND is a generalization of Alt-GDA, which has been shown to outperform GDA numerically in some cases [30]. We analyze its convergence in Section IV-C, and compare its numerical performance with GRAND in Section VI-C.

Alt-GRAND allows for scaled implementations if $s^k = P^k \nabla_x L(x^k, y^k)$ and $t^k = Q^k \nabla_y L(x^{k+1}, y^k)$ with positive definite matrices P^k and Q^k satisfying $\sqrt{\gamma_s \Gamma_s} I_d \preceq P^k \preceq \Gamma_s I_d$ and $\sqrt{\gamma_\tau \Gamma_\tau} I_p \preceq Q^k \preceq \Gamma_\tau I_p$. Newton-type methods are also covered by Alt-GRAND. For example, GDN [22] is a special case when $P^k = \nabla^2_{xx} L(x^k, y^k)$ and $Q^k = I_p$, which has a provable local linear rate. The alternating Newton-type method (Alt-NDA), on the other hand, takes $P^k = [\nabla^2_{xx} L(x^k, y^k)]^{-1}$ and $Q^k = [N(x^{k+1}, y^k)]^{-1}$, similar to NDA. Assumption III.2 holds for t^k under similar conditions as in NDA. Alt-NDA, also known as the complete Newton method [22], has a provable local quadratic rate. We will discuss further the local performance of Alt-NDA in Section V-A.

IV. Global Convergence Analysis

In this section, we analyze the global convergence of GRAND. Theorem IV.5 establishes the linear convergence of GRAND under certain strongly-convex-PL conditions, which ensures the linear rate of the distributed hybrid methods.

A. Preliminaries

We first introduce assumptions and definitions used throughout the section, starting with the standard conditions for L.

Assumption IV.1: The function $L(x, y)$ satisfies that,

(a) L is twice differentiable in (x, y). Its partial gradient $\nabla_x L$ is continuously differentiable relative to (x, y);

(b) Given any $y \in \mathbb{R}^p$, $L(\cdot, y)$ is m_{x}-strongly convex with respect to x with $m_{\mathsf{x}} > 0$;

(c) The partial gradient $\nabla_x L$ is ℓ_{xx}- and ℓ_{xy}-Lipschitz continuous in x and y, respectively. Moreover, $\nabla_y L$ is ℓ_{yx}- and ℓ_{yy}-Lipschitz continuous in x and y, respectively. Here, constants $\ell_{\mathsf{xx}} > m_{\mathsf{x}} > 0$, and $\ell_{\mathsf{xy}}, \ell_{\mathsf{yx}}, \ell_{\mathsf{yy}} \geq 0$.

It is easy to check that L^{dc} defined in (II.3) under Assumption II.1 satisfies Assumption IV.1 with $m_{\mathsf{x}} = m_{\mathsf{dc}}$, $\ell_{\mathsf{xx}} = \ell_{\mathsf{dc}} + 2\mu$, $\ell_{\mathsf{xy}} = \ell_{\mathsf{yx}} = 2$, and $\ell_{\mathsf{yy}} = 0$. Moreover, L^{nf} defined in (II.11) under Assumption II.6 satisfies Assumption IV.1 with $m_{\mathsf{x}} = m_{\mathsf{nf}}$, $\ell_{\mathsf{xx}} = \ell_{\mathsf{nf}}$, $\ell_{\mathsf{xy}} = \ell_{\mathsf{yx}} = \|E\|$, and $\ell_{\mathsf{yy}} = 0$.

Most existing analyses of GDA in strongly-convex-concave settings study linear combinations of $\|x^k - x^\star\|^2$ and $\|y^k - y^\star\|^2$ as Lyapunov functions [17], where $(x^\star, y^\star)$ is a solution to Problem I.1. Let $z^\star = (x^\star; y^\star)$ and $z^k = (x^k; y^k)$. The gradient steps in GDA decrease the Lyapunov function by a ratio such that $\|z^{k+1} - z^\star\|_V^2 \leq \rho \|z^k - z^\star\|_V^2$ for a matrix $V \succeq 0$ and a constant $0 < \rho < 1$ at iteration k, implying a linear rate. However, such analysis does not apply to GRAND due to the time-varying angles between the updates and the gradients. A similar procedure leads to $\|z^{k+1} - z^\star\|_{V^k}^2 \leq \rho \|z^k - z^\star\|_{V^k}^2$ with time-varying matrices $\{V^k \succeq 0\}_k$, which does not ensure convergence. Moreover, such Lyapunov functions also fail in nonconcave cases. Thus, recalling $x^*(y)$ and $\psi(y)$ defined in (II.4), we introduce two performance metrics, y's optimality measure and x's tracking error, with $\Xi_\psi = \max_y \psi(y)$,

$$\begin{aligned} \Delta_y^k &= \Xi_\psi - \psi(y^k), \\ \Delta_x^k &= L(x^k, y^k) - L(x^*(y^k), y^k). \end{aligned} \tag{IV.1}$$

We remark that Δ_x^k and Δ_y^k are nonnegative by definition. Here Δ_y^k measures the distance between y's current function value to its upper bound, and Δ_x^k tracks the error of x's current function value to the optimal one at the current y^k point. We take $\Xi_\psi = \psi(y^\star)$ when ψ has a maximizer $y^\star$. In this case, Δ_y^k is y's optimality gap and becomes zero at the optimal point $(x^\star, y^\star) = (x^*(y^\star), y^\star)$. We will define Lyapunov functions as linear combinations of these performance metrics.

B. Global Convergence of GRAND

This section analyzes the global convergence of GRAND under Assumption III.1. We first define some constants used in the analysis. When $\gamma_t < \Gamma_t$, let $\nu = 1/\sqrt[3]{1 - \gamma_t^2/\Gamma_t^2} - 1 > 0$. We define positive constants c_1, ℓ_ψ, ι, and c_2 as follows,

$$\begin{aligned} c_1 &= (\Gamma_t^2/2\gamma_t)[\nu \mathbb{I}_{\{\gamma_t < \Gamma_t\}}/(1+\nu) + \mathbb{I}_{\{\gamma_t = \Gamma_t\}}], \\ \ell_\psi &= \ell_{\mathsf{yy}} + \ell_{\mathsf{yx}}\ell_{\mathsf{xy}}/m_{\mathsf{x}}, \quad \iota = 2\Gamma_t + \Gamma_t^2 + c_1 \ell_{\mathsf{yy}}/(3\ell_\psi), \\ c_2 &= (\Gamma_t^2/2\gamma_t)\{[1/(1+\nu) + 1 + \nu]\mathbb{I}_{\{\gamma_t < \Gamma_t\}}/\nu + \mathbb{I}_{\{\gamma_t = \Gamma_t\}}\}. \end{aligned} \tag{IV.2}$$

Then we define functions Υ^k and Δ^k as combinations of Δ_y^k and Δ_x^k defined in (IV.1). For $k = 0, 1, \ldots, K$, we have

$$\begin{aligned} \Upsilon^k &= \beta\iota \|\nabla\psi(y^k)\|^2 + (2\alpha\gamma_s m_{\mathsf{x}}/3)\Delta_x^k, \\ \Delta^k &= (3\iota/c_1)\Delta_y^k + \Delta_x^k, \end{aligned} \tag{IV.3}$$

We remark that Υ^k and Δ^k are nonnegative. The function Υ^k is the Lyapunov function measuring the performance of GRAND for strongly-convex-nonconcave problems, while Δ^k is the Lyapunov function in the strongly-convex-PL setting.

1) Strongly-Convex-Nonconcave Settings: We present the sublinear convergence of GRAND in the following theorem.

Theorem IV.2 (Strongly-Convex-Nonconcave): Under Assumptions III.1 and IV.1, with constants c_1, c_2, ι, and ℓ_ψ defined in (IV.2), suppose the stepsizes satisfy $\alpha \leq 2\gamma_s/\{\Gamma_s^2[3\ell_{\mathsf{xx}} + c_1\ell_{\mathsf{yx}}^2/(\ell_\psi\Gamma_t^2)]\}$ and $\beta \leq \min\{c_1/3\ell_\psi\Gamma_t^2, \alpha\gamma_s m_{\mathsf{x}}^2 c_1/[3\ell_{\mathsf{yx}}^2\iota(3c_2 + 2c_1)]\}$. Then the iterates from Algorithm 3 satisfy

$$\left(\sum_{k=0}^{K-1} \Upsilon^k\right)/K \leq \Delta^0/K.$$

Proof Sketch of Theorem IV.2: We decompose our analysis into four steps. *Step 1:* Preparation. *Step 2:* We bound y's optimality measure Δ_y^{k+1} with x's tracking error measured in $\|\nabla_x L(x^k, y^k)\|^2$ by the Lipschitz continuity of $\nabla\psi$. *Step 3:* We bound x's tracking error Δ_x^{k+1} with y's optimality measure $\|\nabla\psi(y^k)\|$ by the Lipschitz continuity of $\nabla_x L(x, y)$ and $\nabla_y L(x, y)$. *Step 4:* Finally, we take a linear combination of the coupled bounds on Δ_y^{k+1} and Δ_x^{k+1}. □

Remark IV.1: Due to space constraints, we omit the proof here. More details are in Appendix A. Interested readers are referred to [33] for the complete proof. Theorem IV.2 presents the global sublinear convergence of GRAND in strongly-convex-nonconcave settings. To illustrate the result, let $\ell = \ell_{\mathsf{xx}} + \ell_{\mathsf{xy}} + \ell_{xy} + \ell_{\mathsf{yy}}$ and $\kappa = \ell/m_{\mathsf{x}}$. Here ℓ and κ characterize the Lipschitz continuity and the condition number of L, respectively. We note that $\iota = O(1)$, $\alpha = O(1/\ell)$, and $\beta = O(1/(\kappa^2\ell))$ under the conditions in Theorem IV.2. Theorem IV.2 implies that $(\sum_{k=0}^{K-1}\|\nabla\psi(y^k)\|^2)/K \le (\sum_{k=0}^{K-1}\Upsilon^k)/(\beta\iota K) \le \Delta^0/(\beta\iota K)$ by the definition of Υ^k. Thus, we need $K = O(\kappa^2\epsilon^{-2})$ iterations to achieve $\min_{k=0,\ldots,K-1}\{\|\nabla\psi(y^k)\|\} \le \epsilon$. Our iteration complexity and stepsizes all match the state-of-the-art rate for GDA in the same setting [19]. As $\{\Upsilon^k\}_{k\ge 0}$ goes to zero, both $\{\|\nabla\psi(y)\|\}_{k\ge 0}$ and $\{\Delta_x^k\}_{k\ge 0}$ goes to zero, which implies that the iterates converge to a point $(x^*(y^\dagger), y^\dagger)$ with $\nabla\psi(y^\dagger) = 0$. Convergence to a stationary point in y is the best we can obtain for strongly-convex-nonconcave problems.

In general, the theoretical convergence speed of the scaled gradient methods has worse constants than the gradient methods since Γ_s/γ_s and Γ_t/γ_t used in the directions are larger than $\Gamma_s/\gamma_s = \Gamma_t/\gamma_t = 1$ used in gradient methods. But these scaling methods under GRAND can provide not only more flexibility but also faster convergence behaviors in practice. See Section VI for more numerical studies and details.

2) Linear Rates for Strongly-Convex-PL Settings: The preceding result can be strengthened to a linear rate if we further impose the assumption that ψ satisfies the following Polyak-Łojasiewicz (PL) inequality.

Assumption IV.3: For any $y \in \mathbb{R}^p$, the function ψ defined in (II.4) has a global maximizer and $-\psi$ satisfies the PL inequality with a positive constant p_ψ.

Let $\psi^\star$ be the maximum function value. Assumption IV.3 gives that for any y, $\|\nabla\psi(y)\|^2/2 \ge p_\psi(\psi^\star - \psi(y))$. PL inequality is a simple sufficient condition to show a global linear rate for gradient descent method on solving minimization problems [34]. As an example, we next show that Assumption IV.3 can be easily satisfied by distributed computing problems. We introduce a structured problem with $f : \mathbb{R}^d \to \mathbb{R}$, $g : \mathbb{R}^p \to \mathbb{R}$, and $W \in \mathbb{R}^{p\times d}$ as follows,

$$L(x,y) = f(x) + y^\intercal W x - g(y). \tag{IV.4}$$

Example IV.4: In a structured problem of the form (IV.4), if there exists a function $h : \mathbb{R}^d \to \mathbb{R}$ such that $g(y) = h(W^\intercal y)$ and $h(\lambda) + m_h\|\lambda\|^2/2$ is convex with $m_h < 1/\ell_{\mathsf{xx}}$, then Assumption IV.3 holds with $p_\psi = \sigma_{\min}^+(W)(1/\ell_{\mathsf{xx}} - m_h)$.

For Problem DC with L^{dc} defined in (II.3), Example IV.4 holds with $h = g = 0$ and $m_h = 0$. Thus, with $\sigma_{\min}^+(W) = 1-\gamma$ and $\ell_{\mathsf{xx}} = \ell_{\mathsf{dc}} + 2\mu$, we have $p_\psi^{\mathsf{dc}} = (1-\gamma)/(\ell_{\mathsf{dc}} + 2\mu)$. Similarly, for Problem NF with L^{nf} defined in (II.11), we have $\ell_{\mathsf{xx}} = \ell_{\mathsf{nf}}$. Since there exists a feasible solution x^{nf} such that $Ex^{\mathsf{nf}} = \pi$, Example IV.4 holds with $g(y) = \pi^\intercal y = (x^{\mathsf{nf}})^\intercal E^\intercal y$ and $h(\lambda) = (x^{\mathsf{nf}})^\intercal\lambda$ and thus $m_h = 0$. Thus, we obtain $p_\psi^{\mathsf{nf}} = \sigma_{\min}^+(E)/\ell_{\mathsf{nf}}$. In addition to the distributed computing problems, Assumption IV.3 can also be naturally satisfied by cases like strongly-convex-strongly-concave settings. See Examples 4.12 - 4.13 in [33] for more examples and details.

Now we study the global convergence of GRAND for strongly-convex-PL problems under Assumption IV.3. In particular, we take $\Xi_\psi = \psi(y^\star)$ as the exact upper bound and have $\Delta_y^k = \psi(y^\star) - \psi(y^k)$ in (IV.1). We note that $\Delta_y^k = 0$ and $\Delta_x^k = 0$ and thus Δ^k defined in (IV.3) is zero at a global minimax point $(x^k, y^k) = (x^*(y^\star), y^\star)$. For convenience, we define a positive constant δ with c_1 and ι defined in (IV.2),

$$\delta = \min\{2\beta\iota p_\psi c_1/(3\iota + c_1), 2\alpha\gamma_s m_{\mathsf{x}}/3\}. \tag{IV.5}$$

The following theorem states the result, where δ serves as the linear rate coefficient.

Theorem IV.5 (Strongly-Convex-PL): Under Assumptions III.1, IV.1, and IV.3, suppose the stepsizes satisfy the conditions in Theorem IV.2 and additionally, $\beta < (3\iota + c_1)/(2\iota p_\psi c_1)$. For all $k = 0, 1, \ldots, K-1$, the iterates from GRAND satisfy

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

where δ is defined in (IV.5) satisfying $0 < \delta < 1$.

Remark IV.2: Theorem IV.5 presents the global linear (Q-linear) rate of GRAND for strongly-convex-PL problems under Assumption IV.3. As $\{\Delta^k\}$ goes to zero, both $\{\Delta_y^k\}$ and $\{\Delta_x^k\}$ goes to zero. Moreover, if $y^\dagger$ is a unique maximizer of ψ, the theorem ensures the convergence of the iterates to the global minimax point $(x^*(y^\dagger), y^\dagger)$.

We recall that the distributed consensus and the network flow problems mentioned in Example IV.4 satisfy Assumption IV.3. Thus, Theorem IV.5 guarantees the global linear convergence of DISH in Algorithm 1 for solving Problem DC and Algorithm 2 for solving Problem NF. A specialized linear result with tighter coefficients for DISH is presented in [1]. In particular, for the selection of $\mu \ge 0$ in DISH, it is challenging to characterize the impact of μ on the linear coefficient due to its presence in both the numerator and the denominator. However, given that second-order updates use an approximated local Hessian inverse $(\nabla^2 f_i(x) + \mu I_d)^{-1}$, we recommend selecting a relatively small constant μ for strongly convex f_i, to preserve more Hessian information.

We highlight that the assumptions in the theorems are intended to guarantee algorithm convergence and provide theoretical bounds on stepsizes. However, in practice, larger stepsizes are usually used to improve the performance. In addition, in distributed problems like empirical risk minimization, the parameters depend on the specific dataset, partitioned across agents. If agents have i.i.d. data, we can sample local data to estimate the parameters and use them to approximate the global ones. Otherwise, agents can simultaneously estimate their local parameters and run a max consensus with finite termination [35] to agree on the bounds of the stepsizes.

We define $\widetilde{\kappa} = \ell/\min\{m_{\mathsf{x}}, p_\psi\}$ to characterize the condition number of L under Assumptions IV.1 and IV.3. We investigate the rate coefficient $1-\delta$ by substituting upper bounds on α and β and obtain $\delta = O(1/\widetilde{\kappa}^3)$. It is slower than the optimal complexity $O(1/\widetilde{\kappa}^2)$ of GDA in strongly-convex-strongly-concave case [29] since we study a more general strongly-convex-PL setting here. In short, δ depends on the function property $\widetilde{\kappa}$ and update angles Γ_t/γ_t and Γ_s/γ_s. Though the theorem is conservative, relying on the worst case of update direction angles, as experiments show in Section VI, scaling matrices and Newton-type updates can accelerate the numerical performance.

C. Global Convergence of Alt-GRAND

We now present global rates of Alt-GRAND. The analysis follows the same steps as those for GRAND in Section IV-B. We define positive constants $\widetilde{\iota} = 2\Gamma_\tau + c_1\ell_{\mathrm{yy}}/(3\ell_\psi)$ and $\widetilde{\delta} = \min\{2\beta\widetilde{\iota}p_\psi c_1/(3\widetilde{\iota}+c_1), 2\alpha\gamma_s m_{\mathrm{x}}/3\}$, and Lyapunov functions $\widetilde{\Upsilon}^k = \beta\widetilde{\iota}\|\nabla\psi(y^k)\|^2 + 2\alpha\gamma_s m_{\mathrm{x}}\Delta_x^k/3$ and $\widetilde{\Delta}^k = (3\widetilde{\iota}/c_1)\Delta_y^k + \Delta_x^k$ as linear combinations of Δ_x^k and Δ_y^k. The convergence results are shown as follows.

Theorem IV.6 (Strongly-Convex-Nonconcave): We assume the stepsizes to satisfy some conditions that $\alpha = O(1/\ell)$ and $\beta = O(1/(\kappa^2\ell))$. Under Assumptions IV.1 and III.2, the iterates from Alt-GRAND satisfy $(\sum_{k=0}^{K-1}\widetilde{\Upsilon}^k)/K \le \widetilde{\Delta}^0/K$.

Theorem IV.7 (Strongly-Convex-PL): Suppose that the stepsizes satisfy some conditions that $\alpha = O(1/\ell)$ and $\beta = O(1/(\kappa^2\ell))$. It holds that $0 < \widetilde{\delta} < 1$. Moreover, under Assumptions IV.1, III.2, and IV.3, for all $k = 0, 1, \ldots, K-1$, the iterates from Alt-GRAND satisfy $\widetilde{\Delta}^{k+1} \le (1-\widetilde{\delta})\widetilde{\Delta}^k$.

Theorems IV.6 and IV.7 demonstrate the global sublinear and linear convergence rates of Alt-GRAND for strongly-convex-nonconcave and strongly-convex-PL scenarios, respectively. These results are similar to the rates achieved by GRAND in Theorem IV.5, with only differences in the coefficient constants. Though comparisons between the theoretical results may not be straightforward, we will evaluate their numerical performance later. Besides a global rate guarantee, Alt-NDA, the Newton-type method, exhibits local quadratic convergence [22] when $\alpha = \beta = 1$. Further discussion on Newton-based methods and their local higher-order rates will be presented in the following section.

V. Newton-Based Methods and Local Higher-Order Rates

Though we do not obtain superlinear rates for distributed hybrid methods due to approximation errors in the distributed setting, this section discusses related Newton-based methods for solving Problem I.1 in centralized settings, where an exact Newton-based step is feasible. These discussions aim to clarify the limitations and potentials of distributed second-order methods, which may lead to the future development of superlinear distributed methods. We study the local quadratic rates of Alt-NDA and its variants in Section V-A. We also explore a modified Newton's method in Section V-B to achieve local cubic rates by reusing the Hessian inversion computation.

A. Multistep Alt-NDA With Local Quadratic Rates

We define two mappings $X, Y : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d \times \mathbb{R}^p$ using the operator N defined in (II.5),

$$X(x,y) = (x - [\nabla^2_{xx}L(x,y)]^{-1}\nabla_x L(x,y), y),$$
$$Y(x,y) = (x, y + [N(x,y)]^{-1}\nabla_y L(x,y)).$$

Throughout this section, we consider $(x^\dagger, y^\dagger)$ as a first-order stationary point of Problem I.1. We focus on the local performance and assume $[N(x,y)]^{-1}$ is well-defined (not necessarily semi-definite) in a neighborhood around $(x^\dagger, y^\dagger)$. Specifically, we consider Alt-NDA, introduced in Section III-B, with $s^k = [\nabla^2_{xx}L(x^k,y^k)]^{-1}\nabla_x L(x^k,y^k)$ and $t^k = [N(x^{k+1},y^k)]^{-1}\nabla_y L(x^{k+1},y^k)$. Alt-NDA can be represented as the following composite update,

$$(x^{k+1}, y^{k+1}) = Y \circ X(x^k, y^k).$$

The local quadratic rates of Alt-NDA are shown under local Lipschitz Hessian conditions near the stationary point [22]. Here, we introduce a modified Alt-NDA, $U_J = (X)^J \circ Y \circ X$, with additional $J \ge 1$ minimization steps. We show that U_J converges to $(x^\dagger, y^\dagger)$ with at least a quadratic rate. The updates of U_J at iteration k are as follows,

$$\begin{aligned}
(x^{k+1,0}, y^k) &= X(x^k, y^k),\\
(x^{k+1,0}, y^{k+1}) &= Y(x^{k+1,0}, y^k)\\
(x^{k+1,j+1}, y^{k+1}) &= X(x^{k+1,j}, y^{k+1}) \text{ for } j = 0, \ldots, J-1,\\
x^{k+1} &= x^{k+1,J}.
\end{aligned} \tag{V.1}$$

Let $S' \in \mathbb{R}^{n\times n}$ be the Jacobian matrix of a mapping $S\colon \mathbb{R}^n \to \mathbb{R}^n$. The following lemma provides a sufficient condition for the local quadratic convergence of any mapping.

Lemma V.1 (Theorem 10.1.7 in [36]): Let $S : \mathbb{R}^n \to \mathbb{R}^n$ and $z^\dagger$ such that $S(z^\dagger) = z^\dagger$. Suppose that S is continuously differentiable on an open ball $\mathcal{B}(z^\dagger, r) \subset \mathbb{R}^n$ and twice differentiable at $z^\dagger$, and $S'(z^\dagger) = 0$. Then there is an open neighborhood $\mathfrak{N} \subset \mathbb{R}^n$ of $z^\dagger$ such that for any $z^0 \in \mathfrak{N}$, the iterates $\{z^k\}_{k\ge 0}$ generated by $z^{k+1} = S(z^k)$ converge to $z^\dagger$ with at least a quadratic rate.

It is straightforward that Newton's method for minimization problems satisfies the above conditions and thus converges at least quadratically in a local neighborhood. We prove the following theorem of U_J's local quadratic rate based on Lemma V.1.

Theorem V.2: Under Assumption IV.1, it holds for any $J \ge 1$ that $U_J'(x^\dagger, y^\dagger) = X'(x^\dagger, y^\dagger)Y'(x^\dagger, y^\dagger)X'(x^\dagger, y^\dagger) = 0$. Moreover, there is an open neighborhood $\mathfrak{N}_{U_J}$ of $(x^\dagger, y^\dagger)$ such that for any $(x^0, y^0) \in \mathfrak{N}_{U_J}$, the iterates $\{(x^k, y^k)\}_{k\ge 0}$ generated by $(x^{k+1}, y^{k+1}) = U_J(x^k, y^k)$ in (V.1) converges to $(x^\dagger, y^\dagger)$ at least quadratically.

Theorem V.2 shows the local quadratic convergence of $\{(x^k, y^k)\}_{k\ge 0}$ generated by U_J. We note that $U_J'(x^\dagger, y^\dagger) = X'(x^\dagger, y^\dagger)Y'(x^\dagger, y^\dagger)X'(x^\dagger, y^\dagger) = 0$ holds for any $J \ge 1$. Thus, when taking $J = 1$ in U_J, two Newton's steps on x in each iteration are enough to ensure a local quadratic rate.

The assumption that S is locally continuously differentiable is stronger than the local Lipschitz Hessian condition used in [22]. Also, our updates in (V.1) require one more minimization step per iteration compared to Alt-NDA. However, the proof of Theorem V.2 is interesting as it is operator-based and significantly shorter than [22]. Our result also generalizes previous works [20], [21] that have shown superlinear convergence of multistep Newton's update for constrained optimization problems with Lagrangian functions.

However, unfortunately, we lose the superlinear rates for the distributed hybrid methods due to the errors introduced by the distributed approximations. It is important to note that existing methods achieving superlinear convergence are either applicable only in centralized settings or necessitate an additional inner loop during each iteration in distributed settings [23]. The preceding results show that superlinear distributed methods would be possible if an efficient and accurate distributed Newton step solver can be developed.

B. Newton's Method and Its Cubic-Rate Modification

We now recall the standard Newton's method and its local quadratic rate. Let $z = (x; y) \in \mathbb{R}^{d+p}$ be the concatenation of x and y by column, and $\Lambda(z) = (\nabla_x L(x,y); \nabla_y L(x,y)) : \mathbb{R}^{d+p} \to \mathbb{R}^{d+p}$ be a gradient operator. The first-order stationarity gives $\Lambda(z^\dagger) = 0$. Thus, finding a first-order stationary point is equivalent to finding the root $z^\dagger$ of the system. By applying Newton's method to this root finding problem with $\nabla\Lambda(z) = \left(\begin{smallmatrix} \nabla^2_{xx}L(x,y) & \nabla^2_{xy}L(x,y) \\ \nabla^2_{yx}L(x,y) & \nabla^2_{yy}L(x,y)\end{smallmatrix}\right)$, we have the updates,

$$z^{k+1} = z^k - [\nabla\Lambda(z^k)]^{-1}\Lambda(z^k). \tag{V.2}$$

To simplify the notation, let $\nabla^2_{xx}L$ denote $\nabla^2_{xx}L(x,y)$ and similarly for $\nabla^2_{xy}L$, $\nabla^2_{yx}L$, and N. By Schur complement, the inverse $(\nabla\Lambda)^{-1}$ is given by,

$$(\nabla\Lambda)^{-1} = \begin{pmatrix} (\nabla\Lambda)^{-1}_{11} & (\nabla\Lambda)^{-1}_{12} \\ [(\nabla\Lambda)^{-1}_{12}]^\intercal & -N^{-1}\end{pmatrix}, \tag{V.3}$$

where $(\nabla\Lambda)^{-1}_{12} = (\nabla^2_{xx}L)^{-1}(\nabla^2_{xy}L)N^{-1}$ and $(\nabla\Lambda)^{-1}_{11} = (\nabla^2_{xx}L)^{-1} - (\nabla^2_{xx}L)^{-1}(\nabla^2_{xy}L)N^{-1}(\nabla^2_{yx}L)(\nabla^2_{xx}L)^{-1}$. Let $\nabla^2_{xx}L^k$ denote $\nabla^2_{xx}L(x^k, y^k)$, and similarly for $\nabla^2_{xy}L^k$, $\nabla^2_{yx}L^k$, and N^k. By substituting (V.3) to (V.2), we obtain the x and y updates in the standard Newton's method.

We remark that it requires matrix inverses $(\nabla^2_{xx}L^k)^{-1}$ and $(N^k)^{-1}$ in each iteration, which needs the same amount of matrix inverse computation as $(\nabla^2_{xx}L^k)^{-1}$ and $[N(x^{k+1}, y^k)]^{-1}$ required by Alt-NDA, as discussed in Section V-A.

Moreover, NDA introduced in Section III uses $\operatorname{diag}((\nabla^2_{xx}L)^{-1}, -N^{-1})$ as a diagonal approximation of the Hessian inverse $(\nabla\Lambda)^{-1}$ given by (V.3). It leaves out the off-diagonal parts and a complicated multiplication on x's diagonal block. NDA saves computation in terms of multiplications; thus, it might not have a quadratic rate.

As for distributed computing, we note that N^{-1} is the most intractable part when designing hybrid methods in Section II. A distributed approximation of the updates in (V.2) is computationally expensive since N^{-1} is involved in each block of $(\nabla\Lambda)^{-1}$. Thus, we instead consider approximations of NDA with simple distributed implementations for solving Problems DC and NF. The multiple steps of approximations were essential to enable an easy and distributed implementation in Algorithms 1 and 2. However, their errors made it impossible to achieve a local quadratic rate even when all updates are second-order.

Modified Newton's Method with Local Cubic Rates. The most computationally expensive step in implementing Alt-NDA and the standard Newton's method is to calculate the matrix inverses $(\nabla^2_{xx}L)^{-1}$ and $(N)^{-1}$ at each iteration. We now provide a more efficient cubically converging implementation of Newton-type updates by reusing the matrix inverse computation. We modify the Newton's method in (V.2) by reusing the inverse $[\nabla\Lambda(z)]^{-1}$ for two consecutive steps,

$$z^{k+\frac{1}{2}} = z^k - [\nabla\Lambda(z^k)]^{-1}\Lambda(z^k),$$
$$z^{k+1} = z^{k+\frac{1}{2}} - [\nabla\Lambda(z^k)]^{-1}\Lambda(z^{k+\frac{1}{2}}).$$

We only need to compute $[\nabla\Lambda(z^k)]^{-1}$ once per iteration in the above updates, which involves computing the matrix inverses $(\nabla^2_{xx}L^k)^{-1}$ and $(N^k)^{-1}$. Thus, the computational cost is equivalent to that of Alt-NDA and the standard Newton's method in each iteration. We can rewrite the updates as,

$$z^{k+1} = z^k - [\nabla\Lambda(z^k)]^{-1}[\Lambda(z^k) + \Lambda(z^k - [\nabla\Lambda(z^k)]^{-1}\Lambda(z^k))]. \tag{V.4}$$

By substituting $[\nabla\Lambda(z^k)]^{-1}$ in (V.3) to (V.4), we can obtain an update formula for x^{k+1} and y^{k+1}. We omit it here for simplicity. The following theorem shows a local cubic convergence rate of updates in (V.4).

Theorem V.3 (Theorem 10.2.4 in [36]): Suppose that there is an open ball $\mathcal{B}(z^\dagger, \widetilde{r}) \subset \mathbb{R}^n$ and a constant $\ell_\Lambda > 0$ such that $\nabla\Lambda(z)$ satisfies $\|\nabla\Lambda(z) - \nabla\Lambda(z^\dagger)\| \le \ell_\Lambda\|z - z^\dagger\|$ for any $z \in \mathcal{B}(z^\dagger, \widetilde{r})$. Suppose that $\nabla\Lambda(z^\dagger)$ is nonsingular. Then the iterates $\{z^k\}_{k\ge 0}$ converge to $z^\dagger$ with a cubic rate.

Theorem V.3 ensures a much faster rate of updates in (V.4) than Alt-NDA and the standard Newton's method, with the same computational cost in terms of the matrix inverse per iteration. This suggests we reuse the Hessian inverses and implement (V.4) locally to achieve cubic rates in practice.

VI. Numerical Experiments

In this section, we conduct numerical experiments. For all problems and methods, we tune stepsizes and parameters by grid search and select the optimal ones with the minimum number of iterations to reach a predetermined error threshold.

A. Consensus Problems

We implement DISH in Algorithm 1 to solve distributed empirical risk minimization problems. We evaluate all methods on two setups, both with synthetic data. In each setup, we generate the underlying network by the Erdős-Rényi model with n nodes and each edge independently with probability p. Let $\deg_{\max} = \max_{i\in\mathcal{N}}\{\deg(i)\}$ be the largest degree over the network and Z be the consensus matrix with elements $z_{ii} = 1 - \deg(i)/(\deg_{\max}+1)$ for $i \in \mathcal{N}$, $z_{ij} = 1/(\deg_{\max}+1)$ for $\{i,j\} \in \mathcal{E}$, and $z_{ij} = 0$ otherwise. Let $\Theta_i \in \mathbb{R}^{N_i \times d}$ and $v_i \in \mathbb{R}^{N_i}$ be local feature matrix and label vector at agent i, respectively, and $\lambda \ge 0$ be a penalty parameter. There are $N = \sum_{i\in\mathcal{N}} N_i$ amount of data with local dataset size N_i. Let $\omega \in \mathbb{R}^d$ be the decision variable. Here are the two setups.

Setup 1: Distributed Linear Least Squares. We study the problem $\min_\omega[(\sum_{i=1}^n \|\Theta_i\omega - v_i\|^2)/(2N) + \lambda\|\omega\|^2/2]$ with $n = 10$, $p = 0.7$, $d = 5$, $N_i = 50$ for $i \in \mathcal{N}$, and $\lambda = 1$. We

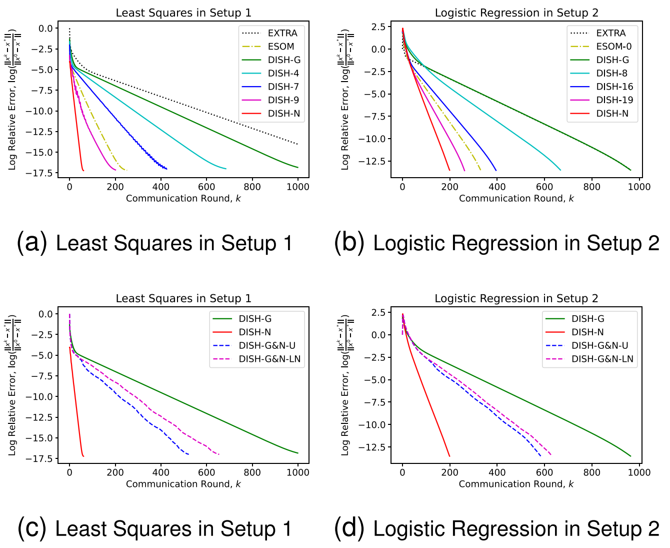


Fig. 1. Results of EXTRA, ESOM-0, DISH, and DISH-G&N.

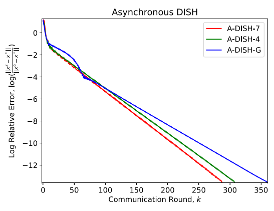


Fig. 2. Asynchronous DISH.

generate features $\widehat{\Theta}_i \in \mathbb{R}^{50\times 5}$, noises $u_i \in \mathbb{R}^{50}$ for $i \in \mathcal{N}$, and $\omega_0 \in \mathbb{R}^5$ from standard Normal distributions. We set $\Theta_i = \widehat{\Theta}_i S$ with a scaling matrix $S = \mathrm{diag}\{10, 10, 0.1, 0.1, 0.1\}$ and generate $v_i \in \mathbb{R}^{50}$ by $v_i = \Theta_i\omega_0 + u_i$ for $i \in \mathcal{N}$.

Setup 2: Distributed Logistic Regression. For $v_i \in \{0,1\}^{N_i}$ and $h_i = 1/(1+\exp(-\Theta_i\omega))$, we study the problem $\min_\omega[(\sum_{i=1}^n [-v_i^\intercal \log h_i - (1-v_i)^\intercal \log(1-h_i)])/N + \lambda\|\omega\|^2/2]$. We set $n = 20$, $p = 0.5$, $d = 3$, $N_i = 50$ for $i \in \mathcal{N}$, and $\lambda = 1$. We generate $\widehat{\Theta}_i \in \mathbb{R}^{50\times 3}$, noises $u_i \in \mathbb{R}^{50}$ for $i \in \mathcal{N}$, and $\omega_0 \in \mathbb{R}^3$ from Normal distributions. We scale $\widehat{\Theta}_i$ with $S = \mathrm{diag}\{10, 0.1, 0.1\}$ and set feature matrices to be $\Theta_i = \widehat{\Theta}_i S$. Moreover, we generate $v_i \in \mathbb{R}^{50}$ by the formula $v_i = \mathrm{argmax}(\mathrm{softmax}(\Theta_i\omega_0 + u_i))$.

We compare EXTRA [7], ESOM-0 [16], and different variants of DISH in Algorithm 1 for the two setups. Let DISH-K represent DISH with K agents consistently performing Newton-type updates while others adopt gradient-type updates. DISH-G &N denote DISH with all agents switching between gradient-type and Newton-type updates occasionally. In particular, DISH-G&N-U and DISH-G&N-LN denote agents changing their update types every t_i iterations, where $t_i \sim U[5, 50]$ and $t_i \sim \mathrm{lognormal}(2, 4) + 30$, respectively. The initial updates for DISH-G&N-U and DISH-G &N-LN are uniformly sampled from {'gradient-type', 'Newton-type'}. The error is measured by $\|x^k - x^\star\|/\|x^0 - x^\star\|$, where $x^\star$ is the optimal solution obtained by a centralized solver. In DISH, we fix $a_i = 1$ for Newton-type updates to mimic the primal Newton's step.

All methods in this study require one communication round with the same communication costs per iteration regardless of the update type. Fig. 1 depicts the number of communication rounds (iterations) on the x-axis and the logarithm of the relative error on the y-axis. The results demonstrate that DISH achieves linear performance regardless of the agents' choice of gradient-type and Newton-type updates, validating the theoretical guarantees presented in Theorem IV.5. Notably, the performance of the first-order methods, such as EXTRA and DISH-G, is similar. However, when some agents adopt Newton-type updates, DISH consistently outperforms the baseline method DISH-G, resulting in faster training. Specifically, DISH-N outperforms ESOM-0 in various scenarios, indicating the benefits of dual Hessian approximation in DISH-N. Additionally, increasing the number of agents performing Newton-type updates (denoted by K) tends to accelerate the convergence of DISH by leveraging more Hessian information. This observation suggests that in practical scenarios, agents with higher computational capabilities or cheaper computation costs can locally implement Newton-type updates to enhance the overall convergence speed of the system.

We conduct additional numerical experiments involving asynchronous updates among agents to illustrate the efficiency of second-order updates even in asynchronous settings. We study Setup 2 with $d = 3$ and $n = 10$. Suppose that there are seven fast agents, two slow agents, and one extremely slow agent. In each communication round, the fast agents can perform 4 first-order updates or 1 second-order update; the two slow agents can perform 1 first-order update each; and the extremely slow agent can perform 0.5 first-order updates, communicating with its neighbors every two rounds. Note that the assumptions for the fast agents, either 4 first-order updates or 1 second-order update, are based on their actual runtime on our machine.[1]

We compare the results for the following three cases. In each communication round, locally, the seven fast agents adopt different types of updates for each case:

(1) A-DISH-7: Each of the seven fast agents performs 1 second-order update;
(2) A-DISH-4: Four of the seven fast agents run 1 second-order update, and the other three run 4 first-order updates;
(3) A-DISH-G: Each of the seven fast agents runs 4 first-order updates.

Fig. 2 shows the results of running the algorithms in the three cases. We observe that our algorithm remains efficient with this asynchronous update scheme. Notably, when performing second-order updates, the fast agents continue to help accelerate the convergence speed of the system.

B. Network Flow Problems

We evaluate the numerical performance of Algorithm 2 on network flow problems. We generate an Erdős-Rényi network with $n = 10$ nodes, where each edge exists with probability $p =$

[1]All the experiments are conducted with Intel Xeon Gold 5218R CPUs.

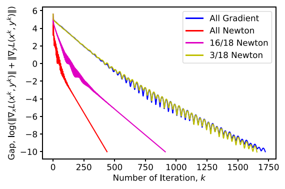


Fig. 3. Network flow problems.

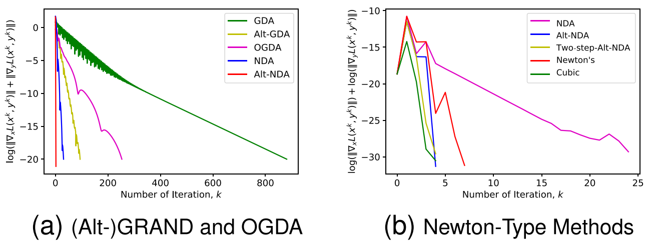


Fig. 4. Centralized minimax problems.

0.4. The resulting connected graph has 18 edges. We study a network flow problem, $\min_x[\sum_{\{i,j\}\in\mathcal{E}}(h_{ij}x_{ij} - v_{ij})^2]/2$ such that $Ex = s$, where $x \in \mathbb{R}^{18}$ is the decision variable, and $E \in \mathbb{R}^{10\times 18}$ is the incidence matrix of the graph. We generate vectors $h \sim \text{lognormal}(3,1), v \sim \text{lognormal}(1,1) \in \mathbb{R}^{18}$, and $\widehat{s} \sim U(0,1) \in \mathbb{R}^{10}$. To ensure feasibility, we set $s_i = \widehat{s}_i - (\sum_{i\in\mathcal{N}} \widehat{s}_i)/n$ so that $\sum_{i\in\mathcal{N}} s_i = 0$.

We evaluate Algorithm 2 with different numbers of Newton-type edges to solve the problem. Specifically, we compare cases where all edges take gradient-type (or Newton-type) updates, and cases where 3 or 16 (out of 18) edges take Newton-type updates. Fig. 3 illustrates the optimal performance of these cases, where the y-axis is the logarithm of the Lagrangian norms. The results show the linear convergence of Algorithm 2 regardless of the agents' update choices. Notably, the all-Newton case exhibits significantly faster convergence, highlighting the effectiveness of the diagonal approximation of the dual Hessian presented in Section II-B3. Additionally, increasing the number of Newton-type agents generally improves the overall performance.

C. Centralized Minimax Problems

We compare the performance of methods on centralized minimax problems. Specifically, we consider GDA and NDA within the GRAND framework and Alt-GDA and Alt-NDA within the Alt-GRAND framework. We also include the optimistic GDA (OGDA) [37] for comparison, which incorporates negative momentum into the gradient updates. Given $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$, $a > 0$, and $\lambda > 0$, we study the strongly-convex-concave problem that $\max_{y\in\mathbb{R}^d}\min_{x\in\mathbb{R}^n}$ $[(\|x\|^2/2 + b^\intercal x + x^\intercal Ay)/n - \lambda R_a(y)]$, where $R_a(y) = \sum_{i=1}^d[\log(1+\exp(ay_i)) + \log(1+\exp(-ay_i))]/a$ is Lipschitz smooth and convex. The problem is a minimax reformulation [17] of the linear regression problem with smoothed-L_1 regularization, $\min_{\omega\in\mathbb{R}^d}[\|A\omega - b\|^2/(2n) + \lambda R_a(\omega)]$. We use a California housing dataset for regression, with $d = 9$, $n = 14448$ (training samples), $a = 10$, and $\lambda = 1/n$. We initialize all methods with $(x^0, y^0) = (0,0)$ and measure optimality using Lagrangian gradient norms.

Fig. 4 shows the optimal performance of the methods. OGDA outperforms GDA. NDA achieves much faster convergence compared to GDA and OGDA due to its utilization of second-order information. The alternating methods, Alt-GDA and Alt-NDA, perform better than their standard counterparts. Note that in GRAND methods, both x and y are updated simultaneously, while in Alt-GRAND methods, y has to wait for the x update. A rough estimate suggests that each iteration of Alt-GRAND takes twice as much time as GRAND. However, our results show that GRAND requires more than two times iterations to converge compared to Alt-GRAND. Thus, Alt-GRAND tends to be more time-efficient overall.

In the second experiment, we compare the local performance of Newton-type methods discussed in Section V. We study the strongly-convex-concave problem $\max_{y\in\mathbb{R}^d}\min_{x\in\mathbb{R}^n}$ $[\|x\|_4^2/2 + b^\intercal x + x^\intercal Ay - \lambda R_a(y)]$, which is a minimax reformulation of the problem $\min_\omega[\|A\omega - b\|_{4/3}^2/2 + \lambda R_a(\omega)]$. We set $n = 5$, $d = 20$, $a = 10$, and $\lambda = 1$, and generate each row of A as $A_i \sim \mathcal{N}(0, I_d)$ and $b \sim \text{lognormal}(0,1)$. We run Newton-type methods, including NDA, Alt-NDA, Twostep-Alt-NDA in (V.1) with $J = 1$, Newton's method, and the cubic method in (V.4). Starting from a point close to the optimal solution obtained by running Alt-GDA for a few rounds, we continue until the gradient norms reach machine precision. We tune the stepsizes in NDA using grid search while fixing the stepsizes in the other methods to be 1. Fig. 4 shows the results of this experiment. NDA exhibits linear rates, while the other methods achieve much faster convergence. The cubic method outperforms the others though the cubic rate is not obvious. Also, the cubic method is more sensitive to the initial point than the others.

VII. Conclusion

This work proposes DISH, a distributed hybrid method that leverages agents' computational heterogeneity. DISH allows agents to choose between gradient-type and Newton-type updates, improving overall efficiency. GRAND is introduced to analyze the performance of methods with general update directions. Theoretical analysis shows global rates for GRAND, ensuring linear convergence for DISH. Future work directions include applying hybrid methods to nonconvex and stochastic settings, exploring hybrid methods with asynchronous updates, and theoretically quantifying the acceleration brought about by the number of second-order agents (K) and the influence of the graph topology.

Appendix A
More Details on the Analysis of Theorem IV.2

We now show more details about the proof of Theorem IV.2. The proofs of the following lemmas and propositions are in [33]. Since x and y-updates are coupled in the descent ascent framework, our idea is to bound y's optimality measure Δ_y^k

and x's tracking error Δ_x^k through coupled inequalities. We decompose our analysis into four steps.

Step 1: Preparation. We start with the Lipschitz continuity of $\nabla\psi(y)$, derived based on the Lipschitz continuity of $x^*(y)$.

Lemma A.1: Under Assumption IV.1, $\nabla\psi(y)$ is ℓ_ψ-Lipschitz continuous with ℓ_ψ defined in (IV.2).

We summarize the properties of update directions s^k and t^k under Assumption III.1.

Lemma A.2: Under Assumption III.1, for any k, it holds that $\gamma_s \le \Gamma_s$ and $\gamma_t \le \Gamma_t$. Moreover,

$$\gamma_s\|\nabla_x L(x^k,y^k)\|^2 \le (s^k)^\intercal \nabla_x L(x^k,y^k),$$
$$\gamma_t\|\nabla_y L(x^k,y^k)\|^2 \le (t^k)^\intercal \nabla_y L(x^k,y^k),$$
$$\|s^k\| \le \Gamma_s\|\nabla_x L(x^k,y^k)\|, \quad \|t^k\| \le \Gamma_t\|\nabla_y L(x^k,y^k)\|.$$

The following holds due to the strong convexity of $L(\cdot,y)$.

Lemma A.3: Under Assumption IV.1, for any k, the iterates from Algorithm 3 satisfy

$$\|\nabla_y L(x^k,y^k) - \nabla\psi(y^k)\| \le (\ell_{\mathsf{yx}}/m_{\mathsf{x}})\|\nabla_x L(x^k,y^k)\|.$$

The next lemma provides an upper bound on the y-update.

Lemma A.4: Under Assumption IV.1, for any $\upsilon > 0$ and any k, the iterates from Algorithm 3 satisfy

$$\|\nabla_y L(x^k,y^k)\|^2 \le (1+\upsilon)\|\nabla\psi(y^k)\|^2 + [(1+1/\upsilon)\ell_{\mathsf{yx}}^2/m_{\mathsf{x}}^2]\|\nabla_x L(x^k,y^k)\|^2.$$

Further with Assumption III.1, it holds for any k that

$$\|y^{k+1}-y^k\|^2 \le 2\beta^2\Gamma_t^2\|\nabla\psi(y^k)\|^2 + (2\beta^2\Gamma_t^2\ell_{\mathsf{yx}}^2/m_{\mathsf{x}}^2)\|\nabla_x L(x^k,y^k)\|^2.$$

Step 2: Bounding y's Optimality Measure Δ_y^{k+1}. We first bound y's updated optimality measure Δ_y^{k+1} with x's tracking error measured in $\|\nabla_x L(x^k,y^k)\|^2$. The analysis follows from the ℓ_ψ-Lipschitz continuity of $\nabla\psi$ in Lemma A.1. The following proposition shows the obtained upper bound on Δ_y^{k+1}.

Proposition A.5: Under Assumptions III.1 and IV.1, with constants c_1, c_2, and ℓ_ψ defined in (IV.2), for all $k = 0,1,\ldots,K-1$, the iterates from Algorithm 3 satisfy $\Delta_y^{k+1} \le \Delta_y^k - (c_1 - \beta\ell_\psi\Gamma_t^2)\beta\|\nabla\psi(y^k)\|^2 + [(c_2+\beta\ell_\psi\Gamma_t^2)\beta\ell_{\mathsf{yx}}^2/m_{\mathsf{x}}^2]\| \nabla_x L(x^k,y^k)\|^2$.

We remark that by checking the derivatives, constants c_1 and c_2 are monotonically increasing and decreasing relative to the ratio γ_t/Γ_t, respectively. It implies that if the ascent direction t^k lies in a smaller angle to $\nabla_y L(x^k,y^k)$ (as γ_t gets closer to Γ_t), c_1 gets larger while c_2 gets smaller. In the extreme case when $t^k = \nabla_y L(x^k,y^k)$ ($\gamma_t = \Gamma_t = 1$), we have $c_1 = c_2 = 1/2$. This provides the tightest upper bound on Δ_y^{k+1} in Proposition A.5 compared to other update directions.

Step 3: Bounding x's Tracking Error Δ_x^{k+1}. Next, we bound x's updated tracking error Δ_x^{k+1} with y's optimality measure $\|\nabla\psi(y^k)\|$. We define constants $\iota_1 = \Gamma_t(2+\Gamma_t+\beta\ell_{\mathsf{yy}}\Gamma_t)$ and $\iota_2 = \alpha[\gamma_s - \alpha\Gamma_s^2(\ell_{\mathsf{xx}}+\beta\ell_{\mathsf{yx}}^2)/2] - \beta\iota_1\ell_{\mathsf{yx}}^2/m_{\mathsf{x}}^2$. The conditions of α and β in Theorem IV.2 ensures $\iota_2 > 0$. The Lipschitz continuity of $\nabla_x L(x,y)$ in x and $\nabla_y L(x,y)$ in x and y gives the following result.

Proposition A.6: Under Assumptions III.1 and IV.1, with constants ι_1 and ι_2 defined above, for all $k = 0,1,\ldots,K-1$, the iterates from Algorithm 3 satisfy $\Delta_x^{k+1} \le \Delta_x^k + \beta\iota_1\|\nabla\psi(y^k)\|^2 - \iota_2\|\nabla_x L(x^k,y^k)\|^2 + \Delta_y^k - \Delta_y^{k+1}$.

Step 4: Putting Things Together. We take a linear combination of the coupled inequalities in Propositions A.5 and A.6 and obtain the following result.

Proposition A.7: Under Assumptions IV.1 and III.1, suppose that the stepsizes satisfy conditions in Theorem IV.2. Then for all $k = 0,1,\ldots,K-1$, the iterates from Algorithm 3 satisfy

$$(3\iota/c_1+1)\Delta_y^{k+1} + \Delta_x^{k+1} \le (3\iota/c_1+1)\Delta_y^k - \beta\iota\|\nabla\psi(y^k)\|^2 + (1-2\alpha\gamma_s m_{\mathsf{x}}/3)\Delta_x^k,$$

where constants c_1 and ι are defined in (IV.2).

Finally, we conclude the proof of Theorem IV.2 by substituting Υ^k and Δ^k.

References

[1] X. Niu and E. Wei, "Dish: A distributed hybrid primal-dual optimization framework to utilize system heterogeneity," in *Proc. IEEE 61st Conf. Decis. Control (CDC),* Piscataway, NJ, USA: IEEE Press, 2022, pp. 6503–6510.

[2] X. Niu and E. Wei,, "Grand: A gradient-related ascent and descent algorithmic framework for minimax problems," in *Proc. 58th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2022, pp. 1–2.

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

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

[5] F. Lamnabhi-Lagarrigue et al., "Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges," *Annu. Rev. Control*, vol. 43, pp. 1–64, 2017.

[6] S. Warnat-Herresthal et al., "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, 2021.

[7] 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.

[8] 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.

[9] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1245–1260, 2017.

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

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

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

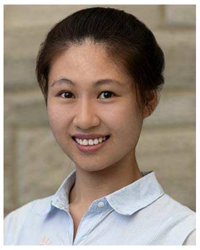
[13] R. Crane and F. Roosta, "Dingo: Distributed Newton-type method for gradient-norm optimization," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[14] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, 1998.

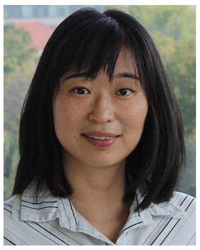
[15] T. Chen et al., "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," in *Proc. Neural Inf. Process. Syst. Workshop Mach. Learn. Syst.*, 2015.

[16] 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, 2016.

[17] 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.*, PMLR, 2019, pp. 196–205.
[18] K. J. Arrow et al., *Studies in Linear and Non-linear Programming*, vol. 2. Stanford, CA, USA: Stanford Univ. Press, 1958.
[19] T. Lin, C. Jin, and M. Jordan, “On gradient descent ascent for nonconvex-concave minimax problems,” in *Int. Conf. Mach. Learn.,* PMLR, 2020, pp. 6083–6093.
[20] 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.
[21] R. H. Byrd, “Local convergence of the diagonalized method of multipliers,” *J. Optim. Theory Appl.*, vol. 26, no. 4, pp. 485–500, 1978.
[22] G. Zhang, K. Wu, P. Poupart, and Y. Yu, “Newton-type methods for minimax optimization,” 2020, *arXiv: 2006.14592*.
[23] A. Mokhtari, Q. Ling, and A. Ribeiro, “Network Newton distributed optimization methods,” *IEEE Trans. Signal Process.*, vol. 65, no. 1, pp. 146–161, Jan. 2016.
[24] 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.
[25] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[26] M. Eisen, A. Mokhtari, and A. Ribeiro, “A primal-dual quasi-newton method for exact consensus optimization,” *IEEE Trans. Signal Process.*, vol. 67, no. 23, pp. 5983–5997, Dec. 2019.
[27] X. Niu and E. Wei, “Fedhybrid: A hybrid federated optimization method for heterogeneous clients,” *IEEE Trans. Signal Process.*, vol. 71, pp. 150–163, 2023.
[28] 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.
[29] Y. Nesterov and L. Scrimali, “Solving strongly monotone variational and quasi-variational inequalities,” *Discrete Continuous Dynamical Syst.*, vol. 31, no. 4, 2011, Art. no. 1383.
[30] J. P. Bailey, G. Gidel, and G. Piliouras, “Finite regret and cycles with fixed step-size via alternating gradient descent-ascent,” in *Proc. Conf. Learn. Theory*, PMLR, 2020, pp. 391–407.
[31] K. Huang, J. Zhang, and S. Zhang, “Cubic regularized Newton method for the saddle point models: A global and local convergence analysis,” *J. Sci. Comput.*, vol. 91, no. 2, pp. 1–31, 2022.
[32] Y. Wang, G. Zhang, and J. Ba, “On solving minimax optimization locally: A follow-the-ridge approach,” in *Proc. Int. Conf. Learn. Representations*, 2019.
[33] X. Niu and E. Wei, “Dish: A distributed hybrid optimization method leveraging system heterogeneity,” 2022, *arXiv: 2212.02638*.
[34] H. Karimi, J. Nutini, and M. Schmidt, “Linear convergence of gradient and proximal-gradient methods under the polyakłojasiewicz condition,” in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, Springer, 2016, pp. 795–811.
[35] F. Iutzeler, P. Ciblat, and J. Jakubowicz, “Analysis of max-consensus algorithms in wireless channels,” *IEEE Trans. Signal Process.*, vol. 60, no. 11, pp. 6103–6107, Nov. 2012.
[36] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. Philadelphia, PA, USA: SIAM, 2000.
[37] A. Mokhtari, A. Ozdaglar, and S. Pattathil, “A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach,” in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2020, pp. 1497–1507.



Xiaochun Niu received the B.S. degree from the Department of Mathematics, Nanjing University, China, in 2019, and the Ph.D. degree from the Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA, in 2024, advised by Professor Ermin Wei. Her research interests include distributed optimization, multiagent learning, and network inference. She is broadly interested in optimization, machine learning, and applied probability, especially with applications in multiagent systems and networks.



Ermin Wei received the undergraduate 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 Massachusetts Institute of Technology, Cambridge, MA, USA, and the Ph.D. degree in electrical engineering and computer science from Massachusetts Institute of Technology, in 2014, advised by Professor Asu Ozdaglar. She is an Associate 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.