Decentralized Quadratically Approximated Alternating Direction Method of Multipliers

Aryan Mokhtari† Wei Shi⋆ Qing Ling⋆ Alejandro Ribeiro†
†Department of Electrical and Systems Engineering, University of Pennsylvania
⋆Department of Automation, University of Science and Technology of China

***Abstract*—This paper considers an optimization problem that components of the objective function are available at different nodes of a network and nodes are allowed to only exchange information with their neighbors. The decentralized alternating method of multipliers (DADMM) is a well-established method for solving this category of problems; however, implementation of DADMM requires solving an optimization subproblem at each iteration for each node. This procedure is often computationally costly for the nodes. We introduce a decentralized quadratic approximation of ADMM (DQM) that reduces computational complexity of DADMM by minimizing a quadratic approximation of the objective function. Notwithstanding that DQM successively minimizes approximations of the cost, it converges to the optimal argument at a linear rate which is identical to the convergence rate of DADMM. Further, we show that as time passes the coefficient of linear convergence for DQM approaches the one for DADMM. Numerical results demonstrate the effectiveness of DQM.**

***Index Terms*—Multi-agent network, decentralized optimization, alternating direction method of multipliers.**

I. Introduction

Decentralized algorithms are designed to solve the problem of minimizing a global cost function over a set of nodes. Agents (nodes) only have access to their local cost functions and try to minimize the global cost cooperatively only by exchanging information with their neighbors. To be more precise, consider a variable $\tilde{\mathbf{x}} \in \mathbb{R}^p$ and a connected network containing n agents each of which has access to a local cost function $f_i : \mathbb{R}^p \to \mathbb{R}$. The agents cooperate to solve the optimization problem

$$\tilde{\mathbf{x}}^* \;=\; \operatorname*{argmin}_{\tilde{\mathbf{x}}} \sum_{i=1}^{n} f_i(\tilde{\mathbf{x}}). \tag{1}$$

Problems of this form arise in decentralized control systems [1]–[3], wireless communication networks [4], [5], sensor networks [6]–[8], and large scale machine learning systems [9]–[11].

There are different algorithms to solve (1) in a distributed manner [6], [12]–[22]. Decentralized implementations of the alternating direction method of multipliers (DADMM) are well-known for solving (1), with a fast linear convergence rate [6], [21], [22]. On the other hand, DADMM steps are computationally costly, since each node has to minimize a convex problem at each iteration. This issue is addressed by the Decentralized Linearized ADMM (DLM) algorithm that in lieu of minimizing the exact primal convex problem, minimizes a first-order linearized version of the primal objective function [23], [24]. DLM reduces the computational complexity of DADMM, however, convergence rate of DLM is slow when the primal objective function is ill-conditioned, since DLM operates on first-order information. Moreover, the coefficient of linear convergence for DLM is strictly smaller than the one for DADMM [25]. These drawbacks can be resolved by incorporating second-order information of the primal function.

Supported by NSF CAREER CCF-0952867 and ONR N00014-12-1-0997.

In this paper we propose a decentralized quadratic approximation of ADMM (DQM) that successively minimizes a quadratic approximation of the primal objective function. Therefore, DQM enjoys the low computation cost of DLM, while its convergence is exact and linear same as DADMM. This approximation incorporates the second-order information of the primal objective function which leads to a provably fast linear convergence even in the case that the cost function is ill-conditioned. Further, we show that as time progresses the coefficient of linear convergence for DQM approaches the one for DADMM.

We begin the paper with describing the DADMM steps and explaining the idea of the DLM algorithm (Section II). Then, we introduce the DQM method which is different from DADMM and DLM in minimizing a quadratic approximation of the primal function (Section III). We analyze convergence properties of DQM (Section IV) and evaluate its performance in solving a logistic regression problem (Section V). Proofs of the results in this paper are provided in [25].

II. DADMM: Decentralized alternating direction method of multipliers

Consider a connected network with n nodes and m edges where the set of nodes is $\mathcal{V} = \{1, \ldots, n\}$ and the set of ordered edges $\mathcal{E}$ contains pairs of nodes (i, j). Node i can communicate with node j if and only if the pair $(i, j) \in \mathcal{E}$ and we further assume that the network is symmetric so that $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$. We define the neighborhood of node i as the set $\mathcal{N}_i = \{j \mid (i, j) \in \mathcal{E}\}$ of nodes that can communicate with i. In problem (1) agent i has access to the local objective function $f_i(\tilde{\mathbf{x}})$ and agents cooperate to minimize the global cost $\sum_{i=1}^{n} f_i(\tilde{\mathbf{x}})$. This specification is more naturally formulated by defining variables $\mathbf{x}_i$ representing the local copies of the variable $\tilde{\mathbf{x}}$. In DADMM we further introduce the auxiliary variables $\mathbf{z}_{ij}$ associated with edge $(i, j) \in \mathcal{E}$ and rewrite (1) as

$$\{\mathbf{x}_i^*\}_{i=1}^n \;:=\; \operatorname*{argmin}_{\mathbf{x}} \sum_{i=1}^{n} f_i(\mathbf{x}_i), \quad \text{s.t. } \mathbf{x}_i = \mathbf{z}_{ij},\ \mathbf{x}_j = \mathbf{z}_{ij}, \quad \text{for all } (i, j) \in \mathcal{E}. \tag{2}$$

The constraints $\mathbf{x}_i = \mathbf{z}_{ij}$ and $\mathbf{x}_j = \mathbf{z}_{ij}$ enforce consensus among neighbors and, since the network is connected, they also enforce global consensus. Therefore, problems (1) and (2) are equivalent in the sense that for all i and j the optimal arguments of (2) satisfy $\mathbf{x}_i^* = \tilde{\mathbf{x}}^*$ for all $i \in \mathcal{V}$ and $\mathbf{z}_{ij} = \tilde{\mathbf{x}}^*$, for all $(i, j) \in \mathcal{E}$.

To specify the DADMM algorithm we write the constraints of (2) in matrix form. Begin by defining the block source matrix $\mathbf{A}_s \in \mathbb{R}^{mp\times np}$ which contains $m \times n$ square blocks $(\mathbf{A}_s)_{e,i} \in \mathbb{R}^{p\times p}$. The block $(\mathbf{A}_s)_{e,i}$ is not identically null if and only if the edge $e = (i, j)$ corresponds to $(i, j) \in \mathcal{E}$ in which case $(\mathbf{A}_s)_{e,i} = \mathbf{I}_p$. Likewise, define the block destination matrix $\mathbf{A}_d \in \mathbb{R}^{mp\times np}$ containing $m \times n$ square blocks $(\mathbf{A}_d)_{e,i} \in \mathbb{R}^{p\times p}$. The square block $(\mathbf{A}_d)_{e,i}$ is equal to identity matrix $\mathbf{I}_p$ if and only if $e = (j, i)$ corresponds to $(j, i) \in \mathcal{E}$, otherwise the block is null. Further define the vector $\mathbf{x} := [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{np}$ as the vector that concatenates the local variables $\mathbf{x}_i$ and the vector $\mathbf{z} := [\mathbf{z}_1, \dots, \mathbf{z}_m] \in \mathbb{R}^{mp}$ that concatenates the auxiliary variables $\mathbf{z}_e = \mathbf{z}_{ij}$. If we further define the aggregate function $f : \mathbb{R}^{np} \to \mathbb{R}$ as $f(\mathbf{x}) := \sum_{i=1}^n f_i(\mathbf{x}_i)$ we can rewrite (2) as

$$\min_{\mathbf{x}} f(\mathbf{x}), \qquad \text{s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{0}. \tag{3}$$

where the matrices $\mathbf{A} \in \mathbb{R}^{2mp\times np}$ and $\mathbf{B} \in \mathbb{R}^{2mp\times mp}$ are defined as the stacks $\mathbf{A} := [\mathbf{A}_s; \mathbf{A}_d]$ and $\mathbf{B} := [-\mathbf{I}_{mp}; -\mathbf{I}_{mp}]$.

Introduce now multipliers $\boldsymbol{\alpha}_e = \boldsymbol{\alpha}_{ij}$ and $\boldsymbol{\beta}_e = \boldsymbol{\beta}_{ij}$ respectively associated with the constraints $\mathbf{x}_i = \mathbf{z}_{ij}$ and $\mathbf{x}_j = \mathbf{z}_{ij}$ [cf. (2)]. Concatenate the multipliers $\boldsymbol{\alpha}_e$ in the vector $\boldsymbol{\alpha} := [\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_m]$ and, likewise, stack the Lagrange multipliers $\boldsymbol{\beta}_e$ in the vector $\boldsymbol{\beta} := [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_m]$. Further stacking the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ into $\boldsymbol{\lambda} := [\boldsymbol{\alpha}; \boldsymbol{\beta}] \in \mathbb{R}^{2mp}$ leads to the Lagrange multiplier $\boldsymbol{\lambda}$ associated with the constraint $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{0}$ in (3). The augmented Lagrangian of (3), and (2), which is the same optimization problem written with different notation, is now defined as

$$\mathcal{L}(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) := f(\mathbf{x}) + \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}) + \frac{c}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}\|^2, \tag{4}$$

where $c > 0$ is an arbitrary positive constant.

The idea of the DADMM algorithm is to alternatively minimize the Lagrangian $\mathcal{L}(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda})$ with respect to the variables $\mathbf{x}$ and the auxiliary variable $\mathbf{z}$ and to follow these minimizations by a multiplier update collinear with the constraint violation. Specifically, consider a time index $k \in \mathbb{N}$ and define $\mathbf{x}_k$, $\mathbf{z}_k$, and $\boldsymbol{\lambda}_k$ as the primal and dual iterates at step k. The first step of DADMM is Lagrangian minimization with respect to $\mathbf{x}$ with $\mathbf{z}_k$ and $\boldsymbol{\lambda}_k$ given,

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{\lambda}_k^T (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k) + \frac{c}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k\|^2. \tag{5}$$

The second step is minimization with respect to the auxiliary variable $\mathbf{z}$ with $\boldsymbol{\lambda}_k$ given but using the updated variable $\mathbf{x}_{k+1}$,

$$\mathbf{z}_{k+1} = \operatorname*{argmin}_{\mathbf{z}} f(\mathbf{x}_{k+1}) + \boldsymbol{\lambda}_k^T (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}) + \frac{c}{2} \|\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}\|^2. \tag{6}$$

With the primal iterates $\mathbf{x}_{k+1}$ and $\mathbf{z}_{k+1}$ updated, the third and final step is to move the Lagrange multiplier $\boldsymbol{\lambda}_k$ in the direction of the constraint violation $\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1}$,

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + c (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1}), \tag{7}$$

where the constant c is the same constant used in (4). The DADMM algorithm follows from the observation that the updates in (5)-(7) can be implemented in a distributed manner, [6], [21], [22].

Notice that the update formulas in (6) and (7) have low computational cost. However, the update for the primal variable $\mathbf{x}$ in (5) requires solution of an optimization problem. To avoid the cost of this minimization, the primal variable $\mathbf{x}_{k+1}$ can be updated inexactly. This idea leads to the DLM algorithm that approximates the objective function value $f(\mathbf{x}_{k+1})$ in (5) through a linearization of the function f in a neighborhood of the current variable $\mathbf{x}_k$. This approximation is defined as $f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) + \rho\|\mathbf{x} - \mathbf{x}_k\|^2$, where ρ is an arbitrary positive constant. Using this approximation, the update formula for the primal variable $\mathbf{x}$ in DLM replaces (5) by

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}} f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) + \rho\|\mathbf{x} - \mathbf{x}_k\|^2 + \boldsymbol{\lambda}_k^T (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k) + \frac{c}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k\|^2. \tag{8}$$

Since the problem in (8) is quadratic, its solution is elementary. E.g., the first order optimality condition for (8) requires the updated variable $\mathbf{x}_{k+1}$ to satisfy

$$\nabla f(\mathbf{x}_k) + \rho(\mathbf{x}_{k+1} - \mathbf{x}_k) + \mathbf{A}^T\boldsymbol{\lambda}_k + c\mathbf{A}^T (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_k) = \mathbf{0}. \tag{9}$$

The expression in (9) is a linear equation for $\mathbf{x}$ that an be solved by inversion of the positive definite matrix $\rho\mathbf{I} + c\mathbf{A}^T\mathbf{A}$.

The sequence of variables $\mathbf{x}_k$ generated by DLM converges linearly to the optimal argument $\mathbf{x}^*$ [23]. This is the same rate of convergence of DADMM, but the linear rate coefficient of DLM is strictly smaller than the linear rate coefficient of DADMM [?]. In this paper we propose an alternative approximation that will be shown to achieve linear convergence with a coefficient that is asymptotically equivalent to the DADMM coefficient. This approximation utilizes second order information of the local functions $f_i(\mathbf{x})$ and leads to the DQM algorithm that we introduce in the following section.

III. DQM: Decentralized Quadratically Approximated ADMM

We introduce a Decentralized Quadratic Approximation of ADMM (DQM) that uses a quadratic approximation of the primal function $f(\mathbf{x})$ for updating the variable $\mathbf{x}$. To be more precise, instead of using $f(\mathbf{x}_{k+1})$ in the optimization problem (5), DQM executes the quadratic approximation $f(\mathbf{x}_k)+\nabla f(\mathbf{x}_k)^T(\mathbf{x}-\mathbf{x}_k)+(1/2)(\mathbf{x}-\mathbf{x}_k)^T\mathbf{H}_k(\mathbf{x}-\mathbf{x}_k)$ for computing the new iterate $\mathbf{x}_{k+1}$, where $\mathbf{H}_k := \nabla^2 f(\mathbf{x}_k)$ is the Hessian of the primal function f computed at $\mathbf{x}_k$. Therefore, the primal variable $\mathbf{x}$ is updated as

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}} f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T\mathbf{H}_k(\mathbf{x} - \mathbf{x}_k) + \boldsymbol{\lambda}_k^T (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k) + \frac{c}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k\|^2. \tag{10}$$

The DQM updates for the variables $\mathbf{z}_k$ and $\boldsymbol{\lambda}_k$ are identical to the DADMM updates in (6) and (7), respectively.

Comparison of the updates in (8) and (10) shows that in DLM the quadratic term $\rho\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2$ is added to the first-order approximation of the primal objective function, while in DQM the second-order approximation of the primal objective function is used to reach a more accurate approximation for $f(\mathbf{x}_{k+1})$. First order optimality conditions of updates in (10), (6) and (7) imply that the DQM iterates can be generated by solving the equations

$$\begin{aligned} \nabla f(\mathbf{x}_k) + \mathbf{H}_k(\mathbf{x}_{k+1} - \mathbf{x}_k) + \mathbf{A}^T\boldsymbol{\lambda}_k + c\mathbf{A}^T(\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_k) &= \mathbf{0}, \\ \mathbf{B}^T\boldsymbol{\lambda}_k + c\mathbf{B}^T (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1}) &= \mathbf{0}, \\ \boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k - c (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1}) &= \mathbf{0}. \end{aligned} \tag{11}$$

Notice that the first equation in (11) can be solved by inverting $\mathbf{H}_k + c\mathbf{A}^T\mathbf{A}$. To guarantee that the system of equations in (11) can

be solved in a distributed manner, we assume a specific structure for the initial vectors $\boldsymbol{\lambda}_0 = [\boldsymbol{\alpha}_0; \boldsymbol{\beta}_0]$, $\mathbf{x}_0$, and $\mathbf{z}_0$ as mentioned in Assumption 1. Before introducing this assumption we define the oriented incidence matrix as $\mathbf{E}_o := \mathbf{A}_s - \mathbf{A}_d$ and the unoriented incidence matrix as $\mathbf{E}_u := \mathbf{A}_s + \mathbf{A}_d$.

Assumption 1 *The initial Lagrange multipliers $\boldsymbol{\alpha}_0$ and $\boldsymbol{\beta}_0$, and the initial variables $\mathbf{x}_0$ and $\mathbf{z}_0$ are chosen such that*

a) $\boldsymbol{\alpha}_0 = -\boldsymbol{\beta}_0$,
b) $\mathbf{E}_u \mathbf{x}_0 = 2\mathbf{z}_0$,
c) $\boldsymbol{\alpha}_0$ *lies in the column space of* $\mathbf{E}_o$.

Assumption 1 enforces some initial conditions on the Lagrange multipliers $\boldsymbol{\alpha}_0$ and $\boldsymbol{\beta}_0$, and the initial variables $\mathbf{x}_0$ and $\mathbf{z}_0$. These conditions are satisfied by setting $\boldsymbol{\alpha}_0 = \boldsymbol{\beta}_0 = \mathbf{0}$ and $\mathbf{x}_0 = \mathbf{z}_0 = \mathbf{0}$. Considering the initial conditions in Assumption 1 we propose a simpler update rule for generating the iterates $\mathbf{x}_k$ instead of solving equations in (11).

Proposition 1 *Consider the system of equations for the DQM algorithm in* (11) *and define the sequence $\boldsymbol{\phi}_k := \mathbf{E}_o^T \boldsymbol{\alpha}_k$. Further, define the unoriented Laplacian as $\mathbf{L}_u := (1/2)\mathbf{E}_u^T \mathbf{E}_u$, the oriented Laplacian as $\mathbf{L}_o := (1/2)\mathbf{E}_o^T \mathbf{E}_o$, and the Degree matrix as $\mathbf{D} := (\mathbf{L}_u + \mathbf{L}_o)/2$. If Assumption 1 holds true, the DQM variables $\mathbf{x}_k$ can be generated as*

$$\mathbf{x}_{k+1} = (2c\mathbf{D} + \mathbf{H}_k)^{-1} \left[(c\mathbf{L}_u + \mathbf{H}_k)\mathbf{x}_k - \nabla f(\mathbf{x}_k) - \boldsymbol{\phi}_k \right], \quad (12)$$

$$\boldsymbol{\phi}_{k+1} = \boldsymbol{\phi}_k + c\mathbf{L}_o \mathbf{x}_{k+1}. \quad (13)$$

Proposition 1 states that by introducing the new variables $\boldsymbol{\phi}_k$, the update formulas for the DQM iterates can be computed using the primal objective function Hessian $\mathbf{H}_k$, the degree matrix $\mathbf{D}$, and the oriented and unoriented Laplacians $\mathbf{L}_o$ and $\mathbf{L}_u$. This observation guarantees that the updates in (12) and (13) are implementable in a distributed manner, since all of these matrices can be computed using local and neighboring information of the nodes. To be more precise, the matrix $2c\mathbf{D} + \mathbf{H}_k$ is block diagonal and its i-th diagonal block is given by $2cd_i\mathbf{I} + \nabla^2 f_i(\mathbf{x}_i)$ which is locally available at node i. Likewise, the inverse matrix $(2c\mathbf{D} + \mathbf{H}_k)^{-1}$ is block diagonal and locally computable since the i-th diagonal block is $(2cd_i\mathbf{I} + \nabla^2 f_i(\mathbf{x}_i))^{-1}$. Computations of the products $\mathbf{L}_u\mathbf{x}_k$ and $\mathbf{L}_o\mathbf{x}_{k+1}$ can be implemented in a decentralized manner as well, since the Laplacian matrices $\mathbf{L}_u$ and $\mathbf{L}_o$ are block neighbor sparse. Note that a matrix is block neighbor sparse when its ij-th block is not null if and only if nodes i and j are neighbors or $j = i$. Therefore, nodes can compute $\mathbf{L}_u\mathbf{x}_k$ and $\mathbf{L}_o\mathbf{x}_{k+1}$ by exchanging information with their neighbors. By defining the components of $\boldsymbol{\phi}_k$ as $\boldsymbol{\phi}_k := [\boldsymbol{\phi}_{1,k}, \ldots, \boldsymbol{\phi}_{n,k}]$, the update in (12) can be implemented locally as

$$\mathbf{x}_{i,k+1} = \left(2cd_i\mathbf{I} + \nabla^2 f_i(\mathbf{x}_{i,k})\right)^{-1} \Big[cd_i\mathbf{x}_{i,k} + c\sum_{j\in\mathcal{N}_i} \mathbf{x}_{j,k} + \nabla^2 f_i(\mathbf{x}_{i,k})\mathbf{x}_{i,k} - \nabla f_i(\mathbf{x}_{i,k}) - \boldsymbol{\phi}_{i,k} \Big], \quad (14)$$

where $\mathbf{x}_{i,k}$ and $\boldsymbol{\phi}_{i,k}$ are the iterates of node i at step k. Notice that the defintion $\mathbf{L}_u := (1/2)\mathbf{E}_u^T\mathbf{E}_u = (1/2)(\mathbf{A}_s + \mathbf{A}_d)^T(\mathbf{A}_s + \mathbf{A}_d)$ is used to simplify the i-th component of $c\mathbf{L}_u\mathbf{x}_k$ as $cd_i\mathbf{x}_{i,k} + c\sum_{j\in\mathcal{N}_i}\mathbf{x}_{j,k}$. Further, considering the definition $\mathbf{L}_o = (1/2)\mathbf{E}_o^T\mathbf{E}_o = (1/2)(\mathbf{A}_s - \mathbf{A}_d)^T(\mathbf{A}_s - \mathbf{A}_d)$, the i-th component of $c\mathbf{L}_o\mathbf{x}_{k+1}$ can be simplified as $c\sum_{j\in\mathcal{N}_i}(\mathbf{x}_{i,k} - \mathbf{x}_{j,k})$.

Algorithm 1 DQM method at node i

Require: Initial local iterates $\mathbf{x}_{i,0}$ and $\boldsymbol{\phi}_{i,0}$.
1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Update the local iterate $\mathbf{x}_{i,k+1}$

$$\mathbf{x}_{i,k+1} = \left(2cd_i\mathbf{I} + \nabla^2 f_i(\mathbf{x}_{i,k})\right)^{-1} \Big[cd_i\mathbf{x}_{i,k} + c\sum_{j\in\mathcal{N}_i} \mathbf{x}_{j,k} + \nabla^2 f_i(\mathbf{x}_{i,k})\mathbf{x}_{i,k} - \nabla f_i(\mathbf{x}_{i,k}) - \boldsymbol{\phi}_{i,k} \Big].$$

3: Exchange iterates $\mathbf{x}_{i,k+1}$ with neighbors $j \in \mathcal{N}_i$.
4: Update local dual variable $\boldsymbol{\phi}_{k+1}$ as

$$\boldsymbol{\phi}_{i,k+1} = \boldsymbol{\phi}_{i,k} + c\sum_{j\in\mathcal{N}_i} (\mathbf{x}_{i,k+1} - \mathbf{x}_{j,k+1}).$$

5: **end for**

Therefore, the update formula in (13) for node i is given by

$$\boldsymbol{\phi}_{i,k+1} = \boldsymbol{\phi}_{i,k} + c\sum_{j\in\mathcal{N}_i} (\mathbf{x}_{i,k+1} - \mathbf{x}_{j,k+1}). \quad (15)$$

The proposed DQM algorithm is summarized in Algorithm 1. At each iteration k, the primal and dual updates in (14) and (15) are implemented in Steps 2 and 4, respectively. Nodes exchange their local variables $\mathbf{x}_{i,k}$ with their neighbors $j \in \mathcal{N}_i$ in Step 3, since this information is required for Steps 2 and 4.

IV. Convergence Analysis

In this section we show that the sequence of iterates $\mathbf{x}_k$ generated by the DQM method converges linearly to the optimal argument $\mathbf{x}^*$. In addition, we compare the linear convergence coefficients of the DQM and DADMM methods. To provide these results first we make the following assumptions.

Assumption 2 *The local objective functions $f_i(\mathbf{x})$ are twice differentiable and the eigenvalues of the local Hessians $\nabla^2 f_i(\mathbf{x})$ are bounded with positive constants $0 < m \leq M < \infty$. I.e.,*

$$m\mathbf{I} \preceq \nabla^2 f_i(\mathbf{x}) \preceq M\mathbf{I}. \quad (16)$$

Assumption 3 *The local Hessians $\nabla^2 f_i(\mathbf{x})$ are Lipschitz continuous with parameter L. I.e., for all $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^p$, it holds*

$$\left\|\nabla^2 f_i(\mathbf{x}) - \nabla^2 f_i(\hat{\mathbf{x}})\right\| \leq L \|\mathbf{x} - \hat{\mathbf{x}}\|. \quad (17)$$

The lower bound m for the eigenvalues of the local Hessians $\nabla^2 f_i(\mathbf{x})$ implies that the local objective functions $f_i(\mathbf{x})$ are strongly convex with parameter m. The upper bound M for the eigenvalues of the local Hessians $\nabla^2 f_i(\mathbf{x})$ is similar to the condition that the local gradients $\nabla f_i(\mathbf{x})$ are Lipschitz continuous with parameter M. The Lipschitz continuity of the local Hessians imposed by Assumption 3 is typical of second order methods.

Define γ_u and Γ_u as the minimum and maximum singular values of the unoriented incidence matrix $\mathbf{E}_u$, respectively. Further, define γ_o as the smallest non-zero singular value of the oriented incidence matrix $\mathbf{E}_o$. These parameters capture connectivity of the network. Denote the unique solution of (2) as $(\mathbf{x}^*, \mathbf{z}^*)$. Notice that the uniqueness is implied by the strong convexity assumption. Further, define $\boldsymbol{\alpha}^*$ as the unique optimal multiplier that lies in the column space of $\mathbf{E}_o$ – see Lemma 1 of [24] for the uniqueness of such an optimal dual variable $\boldsymbol{\alpha}^*$. We define the energy function $V : \mathbb{R}^{mp\times mp} \to \mathbb{R}$ as introduced in [22] for DADMM,

$$V(\mathbf{z}, \boldsymbol{\alpha}) := c\|\mathbf{z} - \mathbf{z}^*\|^2 + (1/c)\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|^2. \quad (18)$$

The energy function $V(\mathbf{z}, \boldsymbol{\alpha})$ captures the distances of the auxiliary variables $\mathbf{z}_k$ and the dual variables $\boldsymbol{\alpha}_k$ with their optimal arguments $\mathbf{z}^*$ and $\boldsymbol{\alpha}^*$, respectively. Therefore, convergence rate of the energy function is a valid tool for comparing performances of DQM and DADMM. To simplify the notation of the energy function $V(\mathbf{z}, \boldsymbol{\alpha})$, define $\mathbf{u} \in \mathbb{R}^{2mp}$ and $\mathbf{C} \in \mathbb{R}^{2mp \times 2mp}$ as

$$\mathbf{u} := \begin{bmatrix} \mathbf{z} \\ \boldsymbol{\alpha} \end{bmatrix}, \quad \mathbf{C} := \begin{bmatrix} c\mathbf{I} & \mathbf{0} \\ \mathbf{0} & (1/c)\mathbf{I} \end{bmatrix}. \tag{19}$$

The energy function $V(\mathbf{z}, \boldsymbol{\alpha})$ is equal to weighted squared norm $\|\mathbf{u} - \mathbf{u}^*\|_{\mathbf{C}}^2$ where $\mathbf{u}^* := [\mathbf{z}^*; \boldsymbol{\alpha}^*]$. Our goal is to show that the sequence $\|\mathbf{u}_k - \mathbf{u}^*\|_{\mathbf{C}}^2$ converges linearly to null.

Theorem 1 *Consider the DQM method as introduced in* (10)-(15). *Define the sequence of non-negative variables* ζ_k *as*

$$\zeta_k := \min\left\{\frac{L}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|, 2M\right\}. \tag{20}$$

Assume that the constant c *is chosen such that* $c > \zeta_k^2/(m\gamma_u^2)$. *Moreover, consider* $\mu, \mu' > 1$ *as arbitrary constants and* η *as a positive constant chosen from the interval* $(\zeta_k/m, c\gamma_u^2/\zeta_k)$. *If Assumptions 1-3 hold true, then the sequence* $\|\mathbf{u}_k - \mathbf{u}^*\|_{\mathbf{C}}^2$ *generated by DQM satisfies*

$$\|\mathbf{u}_{k+1} - \mathbf{u}^*\|_{\mathbf{C}}^2 \;\leq\; \frac{1}{1+\delta_k}\|\mathbf{u}_k - \mathbf{u}^*\|_{\mathbf{C}}^2\,, \tag{21}$$

where the sequence of positive scalars δ_k *is given by*

$$\delta_k = \min\left\{\frac{c - \eta\zeta_k\gamma_u^{-2}}{\frac{4\mu'\mu\zeta_k^2}{c(\mu'-1)(\mu-1)}\gamma_u^{-2}\gamma_o^{-2} + \frac{\mu'\mu}{(\mu-1)}\Gamma_u^2\gamma_o^{-2}}, \frac{m - \zeta_k/\eta}{\frac{c}{4}\Gamma_u^2 + \frac{\mu}{c}M^2\gamma_o^{-2}}\right\}. \tag{22}$$

Notice that δ_k is a decreasing function of ζ_k and observe that ζ_k is bounded above by $2M$. Therefore, if we substitute ζ_k by $2M$ in (22), the inequality in (21) is still valid. This substitution implies that the sequence $\|\mathbf{u}_k - \mathbf{u}^*\|_{\mathbf{C}}^2$ converges linearly to null. As a result of this convergence we obtain that $\mathbf{u}_k$ approaches the optimal argument $\mathbf{u}^*$. Therefore, the sequence of primal iterates $\mathbf{x}_k$ converges to the optimal argument $\mathbf{x}^*$. This result is formalized in the following corollary.

Corollary 1 *Under the assumptions in Theorem 1, the sequence of squared norms* $\|\mathbf{x}_k - \mathbf{x}^*\|^2$ *generated by the DQM algorithm converges R-linearly to null, i.e.,*

$$\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq \frac{4}{c\gamma_u^2}\|\mathbf{u}_k - \mathbf{u}^*\|_{\mathbf{C}}^2. \tag{23}$$

Corollary 1 states that the sequence $\mathbf{x}_k$ converges to the optimal argument $\mathbf{x}^*$. Hence, we obtain that the sequence $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|$ approaches null. This observation implies that the sequence of scalars ζ_k converges to 0 as time passes, since ζ_k is bounded above by $(L/2)\|\mathbf{x}_{k+1} - \mathbf{x}_k\|$. By considering $\lim_{k\to\infty}\zeta_k = 0$ and making μ' arbitrary close to 1, we obtain

$$\lim_{k\to\infty}\delta_k = \min\left\{\frac{(\mu-1)\gamma_o^2}{\mu\Gamma_u^2}, \frac{m}{\frac{c}{4}\Gamma_u^2 + \frac{\mu}{c}M^2\gamma_o^{-2}}\right\}. \tag{24}$$

Notice that the coefficient δ_k in (24) is identical to the coefficient of linear convergence for the DADMM algorithm [22]. This observation implies that as time passes the coefficient of linear convergence for DQM approaches the one for DADMM.

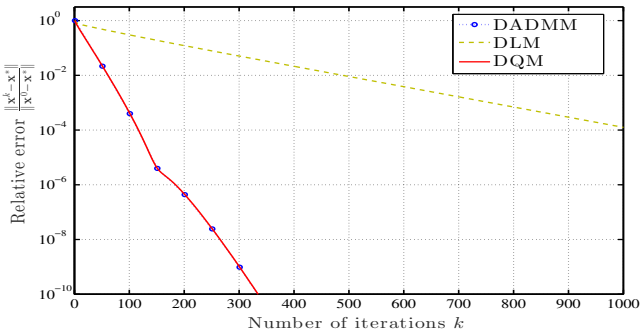

Fig. 1: Relative error $\|\mathbf{x}^k - \mathbf{x}^*\|/\|\mathbf{x}^0 - \mathbf{x}^*\|$ of DADMM, DQM, and DLM versus number of iterations. The convergence path of DQM is almost identical to the one for DADMM, while DQM has less computation complexity. Further, DQM outperforms DLM in convergence speed by orders of magnitude, though at the cost of higher computation complexity.

V. Numerical analysis

In this section we compare performances of DLM, DQM and DADMM in solving a logistic regression problem. We assume that each node in the network has access to q training points. Therefore, the total number of training points is nq. Each of the training samples $\{\mathbf{s}_{il}, y_{il}\}_{l=1}^q$ at node i contains a feature vector $\mathbf{s}_{il} \in \mathbb{R}^p$ with class $y_{il} \in \{-1, 1\}$. It follows from the logistic regression model that the maximum log-likelihood estimate of the classifier $\tilde{\mathbf{x}}$ given the training samples $(\mathbf{s}_{il}, y_{il})$ for $l = 1, \ldots, q$ and $i = 1, \ldots, n$ is

$$\tilde{\mathbf{x}}^* \;:=\; \underset{\tilde{\mathbf{x}}\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n}\sum_{l=1}^{q} \log\Big[1 + \exp(-y_{il}\mathbf{s}_{il}^T\tilde{\mathbf{x}})\Big]. \tag{25}$$

The optimization problem in (25) can be written in the form of (1) by defining the local objective functions f_i as

$$f_i(\tilde{\mathbf{x}}) := \sum_{l=1}^{q} \log\Big[1 + \exp(-y_{il}\mathbf{s}_{il}^T\tilde{\mathbf{x}})\Big]. \tag{26}$$

We compare convergence paths of the DLM, DQM and DADMM algorithms for solving the logistic regression problem. We assume the network contains $n = 10$ nodes and the edges between nodes are generated randomly with probability $P_c = 0.4$. Each agent holds $q = 5$ samples and the dimension of feature vectors is $p = 3$. The reference (ground true) logistic classifier $\tilde{\mathbf{x}}^*$ is pre-computed with a centralized method. Notice that the parameter c for the three methods is optimized by $c_{\text{ADMM}} = 0.7$, $c_{\text{DLM}} = 5.5$, and $c_{\text{DQM}} = 0.7$. Figure 1 illustrates the relative error $\|\mathbf{x}^k - \mathbf{x}^*\|/\|\mathbf{x}^0 - \mathbf{x}^*\|$ of DLM, DQM, and DADMM versus the number of iterations. The convergence path of DQM is almost identical to the convergence path of DADMM and they both converge to the optimal argument faster than DLM. The relative errors $\|\mathbf{x}^k - \mathbf{x}^*\|/\|\mathbf{x}^0 - \mathbf{x}^*\|$ for DQM and DADMM after $k = 300$ iterations are below 10^{-9}, while for DLM the relative error after the same number of iterations is 5×10^{-2}. Conversely, achieving error $\|\mathbf{x}^k - \mathbf{x}^*\|/\|\mathbf{x}^0 - \mathbf{x}^*\| = 10^{-3}$ for DQM and DADMM requires 91 iterations, while DLM requires 758 iterations. Observe that the convergence paths of DQM and DADMM are almost identical, while the computation complexity of DQM is lower than DADMM.

REFERENCES

[1] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

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

[3] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *Signal Processing, IEEE Transactions on*, vol. 56, no. 7, pp. 3122–3136, 2008.

[4] A. Ribeiro, "Ergodic stochastic optimization algorithms for wireless communication and networking," *Signal Processing, IEEE Transactions on*, vol. 58, no. 12, pp. 6369–6386, 2010.

[5] ——, "Optimal resource allocation in wireless communication and networking," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–19, 2012.

[6] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc wsns with noisy links–part i: Distributed estimation of deterministic signals," *Signal Processing, IEEE Transactions on*, vol. 56, no. 1, pp. 350–364, 2008.

[7] U. A. Khan, S. Kar, and J. M. Moura, "Diland: An algorithm for distributed sensor localization with noisy distance measurements," *Signal Processing, IEEE Transactions on*, vol. 58, no. 3, pp. 1940–1947, 2010.

[8] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM, 2004, pp. 20–27.

[9] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[10] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pp. 1543–1550, 2012.

[11] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *Signal Processing Magazine, IEEE*, vol. 31, no. 5, pp. 32–43, 2014.

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

[13] D. Jakovetic, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *Automatic Control, IEEE Transactions on*, vol. 59, no. 5, pp. 1131–1146, 2014.

[14] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *arXiv preprint arXiv:1310.7063*, 2013.

[15] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *arXiv preprint arXiv:1404.6264*, 2014.

[16] A. Mokhtari, Q. Ling, and A. Ribeiro, "An approximate newton method for distributed optimization," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, vol. (to appear). Brisbane, Australia, 2015, available at http://www.seas.upenn.edu/∼aryanm/wiki/NN-ICASSP.pdf.

[17] ——, "Network newton," in *Proc. Asilomar Conf. on Signals Systems Computers*, vol. (to appear). Pacific Grove CA, November 2-5 2014, available at http://arxiv.org/pdf/1412.3740.pdf.

[18] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: convergence analysis and network scaling," *Automatic Control, IEEE Transactions on*, vol. 57, no. 3, pp. 592–606, 2012.

[19] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization." pp. 5453–5458, 2012.

[20] Z.-q. Luo and P. Tseng, "On the convergence rate of dual ascent methods for linearly constrained convex minimization," *Mathematics of Operations Research*, vol. 18, no. 4, pp. 846–867, 1993.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[22] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization," *Signal Processing, IEEE Transactions on*, vol. 62, no. 7, pp. 1750–1761, 2014.

[23] Q. Ling and A. Ribeiro, "Decentralized linearized alternating direction method of multipliers," *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 5447–5451, 2014.

[24] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "Dlm: Decentralized linearized alternating direction method of multipliers," *Signal Processing, IEEE Transactions on*, 2014.

[25] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "Dqm: Decentralized quadratically approximated alternating direction method of multipliers," 2015, available at https://www.seas.upenn.edu/∼aryanm/wiki/DQMjournalpaper.pdf.