AN APPROXIMATE NEWTON METHOD FOR DISTRIBUTED OPTIMIZATION

Aryan Mokhtari[†] *Qing Ling*[⋆] *Alejandro Ribeiro*[†]

[†]Dept. of Electrical and Systems Engineering, University of Pennsylvania
[⋆]Dept. of Automation, University of Science and Technology of China

ABSTRACT

Agents of a network have access to strongly convex local functions f_i and attempt to minimize the aggregate function $f(\mathbf{x}) = \sum_{i=1}^n f_i(\mathbf{x})$ while relying on variable exchanges with neighboring nodes. Various methods to solve this distributed optimization problem exist but they all rely on first order information. This paper introduces Network Newton, a method that incorporates second order information via distributed evaluation of approximations to Newton steps. The method is shown to converge linearly and to do so while exhibiting a quadratic phase. Numerical analyses show substantial reductions in convergence times relative to existing (first order) alternatives.

***Index Terms*—** Multi-agent network, distributed optimization, Newton method

1. INTRODUCTION

Consider a variable $\mathbf{x} \in \mathbb{R}^p$ and a connected network containing n agents each of which has access to a local function $f_i : \mathbb{R}^p \to \mathbb{R}$. The agents cooperate in minimizing the aggregate cost function $f : \mathbb{R}^p \to \mathbb{R}$ taking values $f(\mathbf{x}) := \sum_{i=1}^n f_i(\mathbf{x})$. I.e., agents cooperate in solving the global optimization problem

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

Problems of this form arise often in, e.g., decentralized control systems [1–3], sensor networks [4–6], and large scale machine learning [7–9]. In this paper we assume that the local costs f_i are strongly convex which implies strong convexity of the global cost f. Our goal is to develop an approximate Newton method to solve (1) in distributed settings where agents have access to their local functions only and exchange variables with neighboring agents.

There are different algorithms to solve (1) in a distributed manner. The most popular alternatives are decentralized gradient descent (DGD) [10–13], distributed implementations of the alternating direction method of multipliers [4,14,15], and decentralized dual averaging (DDA) [16,17]. Although there are substantial differences between them, these methods can be generically abstracted as combinations of local descent steps followed by variable exchanges and averaging of information among neighbors. A feature common to these algorithms is that they operate on first order information only. This fact restricts their practical applicability to problems with uncomplicated curvature profiles. Second order methods are not available in general because distributed approximations of Newton steps are difficult to devise. In the particular case of flow optimization problems, these approximations are possible when operating in the dual domain [18–21]. As would be expected, they result in large reductions of convergence times.

The contribution of this paper is to develop Network Newton, a method that relies on distributed approximations of Newton steps to accelerate convergence of the DGD algorithm. The method builds on a reinterpretation of DGD as a linear penalty method to solve (1) (Section 2) – and a Taylor series expansion of the resulting Hessian inverse (Section 3). The method is shown to converge linearly (Section 4) and to do so while exhibiting a quadratic phase (Theorem 2). Numerical analyses show substantial reductions in convergence times relative to DGD (Section 5).

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

2. DECENTRALIZED GRADIENT DESCENT

The network that connects the agents is assumed symmetric and specified by the neighborhoods $\mathcal{N}_i$ that contain the list of nodes than can communicate with i for $i = 1, \ldots, n$. DGD is an established distributed method to solve (1) which relies on the introduction of local variables $\mathbf{x}_i \in \mathbb{R}^p$ and nonnegative weights $w_{ij} \geq 0$ that are not null if and only if $j = i$ or if $j \in \mathcal{N}_i$. Letting $t \in \mathbb{N}$ be a discrete time index and α a given stepsize, DGD is defined by the recursion

$$\mathbf{x}_{i,t+1} = \sum_{j=1}^n w_{ij}\mathbf{x}_{j,t} - \alpha\nabla f_i(\mathbf{x}_{i,t}), \qquad i = 1, \ldots, n. \tag{2}$$

Since $w_{ij} = 0$ when $j \neq i$ and $j \notin \mathcal{N}_i$, it follows from (2) that each agent i updates its estimate $\mathbf{x}_i$ of the optimal vector $\mathbf{x}^*$ by performing an average over the estimates $\mathbf{x}_{j,t}$ of its neighbors $j \in \mathcal{N}_i$ and its own estimate $\mathbf{x}_{i,t}$, and descending through the negative local gradient $-\nabla f_i(\mathbf{x}_{i,t})$. DGD is a distributed method because to implement (2), node i exchanges variables with neighboring nodes only.

It is illuminating to define matrices and vectors so as to rewrite (2) as a single equation. To do so define the vector $\mathbf{y} := [\mathbf{x}_1; \ldots; \mathbf{x}_n] \in \mathbb{R}^{np}$ concatenating the local vectors $\mathbf{x}_i$, as well the vector $\mathbf{h}(\mathbf{y}) := [\nabla f_1(\mathbf{x}_1); \ldots; \nabla f_n(\mathbf{x}_n)] \in \mathbb{R}^{np}$ concatenating the gradients of the local functions f_i taken with respect to the local variable $\mathbf{x}_i$. Further define the matrix $\mathbf{W}$ with entries w_{ij}. It is customary to assume that the weights of a given node sum up to 1 for all i, i.e., $\sum_{j=1}^n w_{ij} = 1$ and that the weights are symmetric, i.e. $w_{ij} = w_{ji}$. If the weights sums up to 1 we must have $\mathbf{W}\mathbf{1} = \mathbf{1}$ which implies that $\mathbf{I} - \mathbf{W}$ is rank deficient. It is also customary to require the rank of $\mathbf{I} - \mathbf{W}$ to be $n - 1$ so that $\text{null}(\mathbf{I} - \mathbf{W}) = \text{span}(\mathbf{1})$. If the two assumptions $\mathbf{W}^T = \mathbf{W}$ and $\text{null}(\mathbf{I} - \mathbf{W}) = \mathbf{1}$ are true, it is possible to show that (2) approaches the solution of (1) in the sense that $\mathbf{x}_{i,t} \approx \mathbf{x}^*$ for all i and large t, [10].

To rewrite (2) define the matrix $\mathbf{Z} := \mathbf{W} \otimes \mathbf{I} \in \mathbb{R}^{np\times np}$ as the Kronecker product of weight matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ and the identity matrix $\mathbf{I} \in \mathbb{R}^{p\times p}$. It is then ready to see that (2) is equivalent to

$$\mathbf{y}_{t+1} = \mathbf{Z}\mathbf{y}_t - \alpha\mathbf{h}(\mathbf{y}_t) = \mathbf{y}_t - \big[(\mathbf{I} - \mathbf{Z})\mathbf{y}_t + \alpha\mathbf{h}(\mathbf{y}_t)\big], \tag{3}$$

where in the second equality we added and subtracted $\mathbf{y}_t$ and regrouped terms. Inspection of (3) reveals that the DGD update formula at step t is equivalent to a (regular) gradient descent algorithm being used to solve the program

$$\mathbf{y}^* := \text{argmin } F(\mathbf{y}) := \min \frac{1}{2}\,\mathbf{y}^T(\mathbf{I} - \mathbf{Z})\,\mathbf{y} + \alpha\sum_{i=1}^n f_i(\mathbf{x}_i). \tag{4}$$

Indeed, just observe that it is possible to write the gradient of $F(\mathbf{y})$ as

$$\mathbf{g}_t := \nabla F(\mathbf{y}_t) = (\mathbf{I} - \mathbf{Z})\mathbf{y}_t + \alpha\mathbf{h}(\mathbf{y}_t), \tag{5}$$

in order to write (3) as $\mathbf{y}_{t+1} = \mathbf{y}_t - \mathbf{g}_t$ and conclude that DGD descends along the negative gradient of $F(\mathbf{y})$ with unit stepsize. The expression

in (2) is just a local implementation of (5) where node i implements the descent $\mathbf{x}_{i,t+1} = \mathbf{x}_{i,t} - \mathbf{g}_{i,t}$ where $\mathbf{g}_{i,t}$ is the ith element of the gradient $\mathbf{g}_t = [\mathbf{g}_{i,t}; \ldots; \mathbf{g}_{i,t}]$. Node i can compute the local gradient $\mathbf{g}_{i,t} = (1 - w_{ii})\mathbf{x}_{i,t} - \sum_{j\in\mathcal{N}_i} w_{ij}\mathbf{x}_{j,t} + \alpha\nabla f_i(\mathbf{x}_{i,t})$ [cf. (2) and (5)] using local information and the $\mathbf{x}_{j,t}$ iterates of its neighbors $j \in \mathcal{N}_i$.

Is it a good idea to descend on $F(\mathbf{y})$ to solve (1)? To some extent. Since we know that the null space of $\mathbf{I} - \mathbf{W}$ is null$(\mathbf{I} - \mathbf{W}) =$ span$(\mathbf{1})$ and that $\mathbf{Z} = \mathbf{W} \otimes \mathbf{I}$ we know that the span of $\mathbf{I} - \mathbf{Z}$ is null$(\mathbf{I} - \mathbf{Z}) =$ span$(\mathbf{1} \otimes \mathbf{I})$. Thus, we have that $(\mathbf{I} - \mathbf{Z})\mathbf{y} = \mathbf{0}$ holds if and only if $\mathbf{x}_1 = \cdots = \mathbf{x}_n$. Since the matrix $\mathbf{I} - \mathbf{Z}$ is positive semidefinite – because it is stochastic and symmetric –, the same is true of the square root matrix $(\mathbf{I} - \mathbf{Z})^{1/2}$. Therefore, we have that the optimization problem in (1) is equivalent to the optimization problem

$$\tilde{\mathbf{y}}^* := \operatorname*{argmin}_{\mathbf{x}} \sum_{i=1}^{n} f_i(\mathbf{x}_i), \qquad \text{s.t.} \quad (\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{y} = \mathbf{0}. \tag{6}$$

Indeed, for $\mathbf{y} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]$ to be feasible in (6) we must have $\mathbf{x}_1 = \cdots = \mathbf{x}_n$ because null$[(\mathbf{I} - \mathbf{Z})^{1/2}] =$ span$(\mathbf{1} \otimes \mathbf{I})$ as already argued. When restricted to this feasible set the objective $\sum_{i=1}^n f_i(\mathbf{x}_i)$ of (6) is the same as the objective of (1) from where it follows that a solution $\tilde{\mathbf{y}}^* = [\tilde{\mathbf{x}}_1^*; \ldots; \tilde{\mathbf{x}}_n^*]$ of (6) is such that $\tilde{\mathbf{x}}_i^* = \tilde{\mathbf{x}}^*$ for all i. The unconstrained minimization in (4) is a penalty version of (6). The penalty function associated with the constraint $(\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{y} = \mathbf{0}$ is the squared norm $(1/2)\|(\mathbf{I} - \mathbf{Z})^{1/2}\mathbf{y}\|^2$ and the corresponding penalty coefficient is $1/\alpha$. Inasmuch as the penalty coefficient $1/\alpha$ is sufficiently large, the optimal arguments $\mathbf{y}^*$ and $\tilde{\mathbf{y}}^*$ are not too far apart.

The reinterpretation of (2) as a penalty method demonstrates that DGD is an algorithm that finds the optimal solution of (4), not (6) or its equivalent (1). To solve (6) we need to introduce a rule to progressively decrease α. Using a fixed α the distance between $\mathbf{y}^*$ and $\tilde{\mathbf{y}}^*$ is of order $O(\alpha)$, [12]. In this paper we exploit the reinterpretation of (3) as a method to minimize (4) to propose an approximate Newton algorithm that can be implemented in a distributed manner. We explain this algorithm in the following section.

3. NETWORK NEWTON

Instead of solving (4) with a gradient descent algorithm as in DGD, we can solve (4) using Newton's method. To implement Newton's method we need to compute the Hessian $\mathbf{H}_t := \nabla^2 F(\mathbf{y}_t)$ of F evaluated at $\mathbf{y}_t$ so as to determine the Newton step $\mathbf{d}_t := \mathbf{H}_t^{-1}\mathbf{g}_t$. Start by differentiating twice in (4) in order to write $\mathbf{H}_t$ as

$$\mathbf{H}_t := \nabla^2 F(\mathbf{y}_t) = \mathbf{I} - \mathbf{Z} + \alpha\mathbf{G}_t, \tag{7}$$

where the matrix $\mathbf{G}_t \in \mathbb{R}^{np\times np}$ is a block diagonal matrix formed by blocks $\mathbf{G}_{ii,t} \in \mathbb{R}^{p\times p}$ containing the Hessian of the ith local function,

$$\mathbf{G}_{ii,t} = \nabla^2 f_i(\mathbf{x}_{i,t}). \tag{8}$$

It follows from (7) and (8) that the Hessian $\mathbf{H}_t$ is block sparse with blocks $\mathbf{H}_{ij,t} \in \mathbb{R}^{p\times p}$ having the sparsity pattern of $\mathbf{Z}$, which is the sparsity pattern of the graph. The diagonal blocks are of the form $\mathbf{H}_{ii,t} = (1 - w_{ii})\mathbf{I} + \alpha\nabla^2 f_i(\mathbf{x}_{i,t})$ and the off diagonal blocks are not null only when $j \in \mathcal{N}_i$ in which case $\mathbf{H}_{ij,t} = w_{ij}\mathbf{I}$.

While the Hessian $\mathbf{H}_t$ is sparse, the inverse $\mathbf{H}_t$ is not. It is the latter that we need to compute the Newton step $\mathbf{d}_t := \mathbf{H}_t^{-1}\mathbf{g}_t$. To overcome this problem we split the diagonal and off diagonal blocks of $\mathbf{H}_t$ and rely on a Taylor's expansion of the inverse. To be precise, write $\mathbf{H}_t = \mathbf{D}_t - \mathbf{B}$ where the matrix $\mathbf{D}_t$ is defined as

$$\mathbf{D}_t := \alpha\mathbf{G}_t + 2\,(\mathbf{I} - \text{diag}(\mathbf{Z})) := \alpha\mathbf{G}_t + 2\,(\mathbf{I} - \mathbf{Z}_d), \tag{9}$$

In the second equality we defined $\mathbf{Z}_d := \text{diag}(\mathbf{Z})$ for future reference. Since the diagonal weights must be $w_{ii} < 1$, the matrix $\mathbf{I} - \mathbf{Z}_d$ is positive definite. The same is true of the block diagonal matrix $\mathbf{G}_t$ because the local functions are assumed strongly convex. Therefore, the matrix $\mathbf{D}_t$ is block diagonal and positive definite. The ith diagonal block $\mathbf{D}_{ii,t} \in \mathbb{R}^p$ of $\mathbf{D}_t$ can be computed and stored by node i as $\mathbf{D}_{ii,t} = \alpha\nabla^2 f_i(\mathbf{x}_{i,t}) + 2(1 - w_{ii})\mathbf{I}$. To have $\mathbf{H}_t = \mathbf{D}_t - \mathbf{B}$ we must define $\mathbf{B} := \mathbf{D}_t - \mathbf{H}_t$. Considering the definitions of $\mathbf{H}_t$ and $\mathbf{D}_t$ in (7) and (9), it follows that

$$\mathbf{B} = \mathbf{I} - 2\mathbf{Z}_d + \mathbf{Z}. \tag{10}$$

Observe that $\mathbf{B}$ is independent of time and depends on the weight matrix $\mathbf{Z}$ only. As in the case of the Hessian $\mathbf{H}_t$, the matrix $\mathbf{B}$ is block sparse with with blocks $\mathbf{B}_{ij} \in \mathbb{R}^{p\times p}$ having the sparsity pattern of $\mathbf{Z}$, which is the sparsity pattern of the graph. Node i can compute the diagonal blocks $\mathbf{B}_{ii} = (1 - w_{ii})\mathbf{I}$ and the off diagonal blocks $\mathbf{B}_{ij} = w_{ij}\mathbf{I}$ using the local information about its own weights only.

Proceed now to factor $\mathbf{D}_t^{1/2}$ from both sides of the splitting relationship to write $\mathbf{H}_t = \mathbf{D}_t^{1/2}(\mathbf{I} - \mathbf{D}_t^{1/2}\mathbf{B}\mathbf{D}_t^{1/2})^{-1}\mathbf{D}_t^{1/2}$. When we consider the Hessian inverse $\mathbf{H}^{-1}$, we can use the Taylor series $(\mathbf{I} - \mathbf{X})^{-1} = \sum_{j=0}^{\infty}\mathbf{X}^j$ with $\mathbf{X} = \mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ to write

$$\mathbf{H}_t^{-1} = \mathbf{D}_t^{-1/2}\sum_{k=0}^{\infty}\left(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\right)^k\,\mathbf{D}_t^{-1/2}. \tag{11}$$

Network Newton (NN) is defined as a family of algorithms that rely on truncations of the series in (11). The Kth member of this family, NN-K considers the first $K+1$ terms of the series to define the approximate Hessian inverse

$$\hat{\mathbf{H}}_t^{(K)^{-1}} := \mathbf{D}_t^{-1/2}\sum_{k=0}^{K}\left(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\right)^k\,\mathbf{D}_t^{-1/2}. \tag{12}$$

NN-K uses the approximate Hessian $\hat{\mathbf{H}}_t^{(K)^{-1}}$ as a curvature correction matrix that is used in lieu of the exact Hessian inverse $\mathbf{H}^{-1}$ to estimate the Newton step. I.e., instead of descending along the Newton step $\mathbf{d}_t := \mathbf{H}_t^{-1}\mathbf{g}_t$ we descend along the NN-K step $\mathbf{d}_t^{(K)} := \hat{\mathbf{H}}_t^{(K)^{-1}}\mathbf{g}_t$, which we intend as an approximation of $\mathbf{d}_t$. Using the explicit expression for $\hat{\mathbf{H}}_t^{(K)^{-1}}$ in (12) we write the NN-K step as

$$\mathbf{d}_t^{(K)} = -\,\mathbf{D}_t^{-1/2}\sum_{k=0}^{K}\left(\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}\right)^k\,\mathbf{D}_t^{-1/2}\,\mathbf{g}_t, \tag{13}$$

where, we recall, the vector $\mathbf{g}_t$ is the gradient of objective function $F(\mathbf{y})$ defined in (5). The NN-K update formula can then be written as

$$\mathbf{y}_{t+1} = \mathbf{y}_t + \epsilon\,\mathbf{d}_t^{(K)}. \tag{14}$$

The algorithm defined by recursive application of (14) can be implemented in a distributed manner because the truncated series in (12) has a local structure controlled by the parameter K. To explain this statement better define the components $\mathbf{d}_{i,t}^{(K)} \in \mathbb{R}^p$ of the NN-K step $\mathbf{d}_t^{(K)} = [\mathbf{d}_{1,t}^{(K)}; \ldots; \mathbf{d}_{n,t}^{(K)}]$. A distributed implementation of (14) requires that node i computes $\mathbf{d}_{i,t}^{(K)}$ so as to implement the local descent $\mathbf{x}_{i,t+1} = \mathbf{x}_{i,t} + \epsilon\mathbf{d}_{i,t}^{(K)}$. The step components $\mathbf{d}_{i,t}^{(K)}$ can be computed through local computations. To see that this is true first note that considering the definition of the NN-K descent direction in (13) the sequence of NN descent directions satisfies

$$\mathbf{d}_t^{(k+1)} = \mathbf{D}_t^{-1}\mathbf{B}\mathbf{d}_t^{(k)} - \mathbf{D}_t^{-1}\mathbf{g}_t = \mathbf{D}_t^{-1}\left(\mathbf{B}\mathbf{d}_t^{(k)} - \mathbf{g}_t\right). \tag{15}$$

Then observe that since the matrix $\hat{\mathbf{B}}$ has the sparsity pattern of the graph, this recursion can be decomposed into local components

$$\mathbf{d}_{i,t}^{(k+1)} = \mathbf{D}_{ii,t}^{-1}\left(\sum_{j\in\mathcal{N}_i, j=i}\mathbf{B}_{ij}\mathbf{d}_{t,j}^{(k)} - \mathbf{g}_{i,t}\right), \tag{16}$$

Algorithm 1 Network Newton-K method at node i

Require: Initial iterate $\mathbf{x}_{i,0}$.

1: **for** $t = 0, 1, 2, \dots$ **do**
2: Exchange iterates $\mathbf{x}_{i,t}$ with neighbors $j \in \mathcal{N}_i$.
3: Gradient: $\mathbf{g}_{i,t} = (1 - w_{ii})\mathbf{x}_{i,t} - \sum_{j \in \mathcal{N}_i} w_{ij}\mathbf{x}_{j,t} + \alpha \nabla f_i(\mathbf{x}_{i,t})$.
4: Compute NN-0 descent direction $\mathbf{d}_{i,t}^{(0)} = -\mathbf{D}_{ii,t}^{-1}\mathbf{g}_{i,t}$
5: **for** $k = 0, \dots, K-1$ **do**
6: Exchange local elements $\mathbf{d}_{i,t}^{(k)}$ of the NN-k step with neighbors
7: NN-$(k+1)$ step: $\mathbf{d}_{i,t}^{(k+1)} = \mathbf{D}_{ii,t}^{-1}\Big(\sum_{j \in \mathcal{N}_i, j=i} \mathbf{B}_{ij}\mathbf{d}_{t,j}^{(k)} - \mathbf{g}_{i,t} \Big)$.
8: **end for**
9: Update local iterate: $\mathbf{x}_{i,t+1} = \mathbf{x}_{i,t} + \epsilon\, \mathbf{d}_{i,t}^{(K)}$.
10: **end for**

The matrix $\mathbf{D}_{ii,t} = \alpha \nabla^2 f_i(\mathbf{x}_{i,t}) + 2(1 - w_{ii})\mathbf{I}$ is stored and computed at node i. The gradient component $\mathbf{g}_{i,t} = (1 - w_{ii})\mathbf{x}_{i,t} - \sum_{j \in \mathcal{N}_i} w_{ij}\mathbf{x}_{j,t} + \alpha \nabla f_i(\mathbf{x}_{i,t})$ is also stored and computed at i. Node i can also evaluate the values of the matrix blocks $\mathbf{B}_{ij} = w_{ij}\mathbf{I}$. Thus, if the NN-$k$ step components $\mathbf{d}_{t,j}^{(k)}$ are available at neighboring nodes j, node i can then determine the NN-$(k+1)$ step component $\mathbf{d}_{i,t}^{(k+1)}$ upon being communicated that information.

The expression in (16) represents an iterative computation embedded inside the NN-K recursion in (14). For each time index t, we compute the local component of the NN-0 step $\mathbf{d}_{i,t}^{(0)} = -\mathbf{D}_{ii,t}^{-1}\mathbf{g}_{i,t}$. Upon exchanging this information with neighbors we use (16) to determine the NN-1 step components $\mathbf{d}_{i,t}^{(1)}$. These can be exchanged and plugged in (16) to compute $\mathbf{d}_{i,t}^{(2)}$. Repeating this procedure K times, nodes ends up having determined their NN-K step component $\mathbf{d}_{i,t}^{(K)}$.

The NN-K method is summarized in Algorithm 1. The descent iteration in (14) is implemented in Step 9. Implementation of this descent requires access to the NN-K descent direction $\mathbf{d}_{i,t}^{(K)}$ which is computed by the loop in steps 4-8. Step 4 initializes the loop by computing the NN-0 step $\mathbf{d}_{i,t}^{(0)} = -\mathbf{D}_{ii,t}^{-1}\mathbf{g}_{i,t}$. The core of the loop is in Step 7 which corresponds to the recursion in (16). Step 6 stands for the variable exchange that is necessary to implement Step 7. After K iterations through this loop the NN-K descent direction $\mathbf{d}_{i,t}^{(K)}$ is computed and can be used in Step 9. Both, steps 4 and 9, require access to the local gradient component $\mathbf{g}_{i,t}$. This is evaluated in Step 3 after receiving the prerequisite information in Step 2.

4. CONVERGENCE ANALYSIS

In this section we show that as time progresses the sequence of objective function $F(\mathbf{y}_t)$ defined in (4) approaches the optimal objective function value $F(\mathbf{y}^*)$. In proving this claim we make the following assumptions.

Assumption 1 *The local objective functions $f_i(\mathbf{x})$ are twice differentiable and the Hessians $\nabla^2 f_i(\mathbf{x})$ have bounded eigenvalues,*

$$m\mathbf{I} \preceq \nabla^2 f_i(\mathbf{x}) \preceq M\mathbf{I}. \tag{17}$$

Assumption 2 *There exists constants $0 \leq \delta < \Delta < 1$ that lower and upper bound the diagonal weights for all i,*

$$0 \leq \delta \leq w_{ii} \leq \Delta < 1 \qquad i = 1, \dots, n. \tag{18}$$

Assumption 3 *The local objective function Hessians $\nabla^2 f_i(\mathbf{x})$ are Lipschitz continuous with parameter L with respect to Euclidian norm, i.e.*

$$\|\nabla^2 f_i(\mathbf{x}) - \nabla^2 f_i(\hat{\mathbf{x}})\| \;\leq\; L\, \|\mathbf{x} - \hat{\mathbf{x}}\|. \tag{19}$$

For the expansion in (11) to be valid the eigenvalues of matrix $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ must be nonnegative and strictly smaller than 1. The following proposition states that this is true for all t – see [22] for this and other proofs.

Proposition 1 *Consider the NN-K method as defined in* (9)-(14)*. If Assumptions 1 and 2 hold true, the matrix $\mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2}$ is positive semidefinite and the eigenvalues are bounded above by a constant $\rho < 1$,*

$$\mathbf{0} \preceq \mathbf{D}_t^{-1/2}\mathbf{B}\mathbf{D}_t^{-1/2} \preceq \rho\mathbf{I} := \frac{2(1-\delta)}{2(1-\delta) + \alpha m}\mathbf{I}. \tag{20}$$

Furthermore, if we define the error of the Hessian inverse approximation as $\mathbf{E}_t^{(K)} = \mathbf{I} - \hat{\mathbf{H}}_t^{(K)^{-1/2}}\mathbf{H}_t\hat{\mathbf{H}}_t^{(K)^{-1/2}}$, the eigenvalues of the error matrix $\mathbf{E}_t^{(K)}$ are bounded as

$$\mathbf{0} \;\preceq \mathbf{E}_t^{(K)} \preceq \rho^{K+1}\mathbf{I}. \tag{21}$$

The eigenvalue upper bound in (20) guarantees that the series in (11) converges. Proposition 1 further asserts that the error in the approximation of the Hessian inverse – thereby on the approximation of the Newton step – is bounded by ρ^{K+1}. This result corroborates the intuition that the larger K is, the closer that $\mathbf{d}_{i,t}^{(K)}$ approximates the Newton step. This closer approximation comes at the cost of increasing the communication cost of each descent iteration. The decrease of this error being proportional to ρ^{K+1} hints that using $N = 1$ or $N = 2$ should provide good enough approximations. This is true in practice – see Section 5. To decrease ρ we can increase δ or α. Increasing δ calls for assigning substantial weight to w_{ii}. Increasing α comes at the cost of moving the solution of (4) away from the solution of (6) and its equivalent (1).

The bounds in Proposition 1 guarantee that the step $\mathbf{d}_t^{(K)}$ in (13) is a descent direction. In turn, this guarantees that with appropriate selection of the stepsize ϵ the sequence of objective function values $F(\mathbf{y}_t)$ generated by NN-K converges to the optimal objective function $F(\mathbf{y}^*)$. This is stated in the following theorem.

Theorem 1 *Consider the NN-K method as defined in* (9)-(14) *and the objective function $F(\mathbf{y})$ as introduced in* (4)*. If the step-size ϵ satisfies*

$$\epsilon = \min\left\{1, \left[\frac{3m(2(1-\Delta)+\alpha m)^3}{L(K+1)^3(2+\alpha M)^{\frac{5}{2}}(F(\mathbf{y}_0) - F(\mathbf{y}^*))^{\frac{1}{2}}}\right]^{\frac{1}{2}}\right\} \tag{22}$$

and Assumptions 1, 2, and 3 hold true, the sequence $F(\mathbf{y}_t)$ converges to the optimal argument $F(\mathbf{y}^*)$ at least linearly with constant $1 - \zeta$. I.e.,

$$F(\mathbf{y}_t) - F(\mathbf{y}^*) \leq (1-\zeta)^t (F(\mathbf{y}_0) - F(\mathbf{y}^*)), \tag{23}$$

where the constant $0 < \zeta < 1$ is explicitly given by

$$\zeta := \frac{(2-\epsilon)\epsilon\alpha m}{2(1-\delta)+\alpha M} - \frac{\epsilon^3 \alpha L(K+1)^3(2+\alpha M)^{\frac{3}{2}}(F(\mathbf{y}_0) - F(\mathbf{y}^*))^{\frac{1}{2}}}{6(2(1-\Delta)+\alpha m)^3}. \tag{24}$$

Theorem 1 shows that the objective function error sequence $F(\mathbf{y}_t) - F(\mathbf{y}^*)$ asymptoticly converges to zero and that the rate of convergence is at least linear. Note that according to the definition of the convergence parameter ζ in Theorem 1, increasing α leads to faster convergence. This observation verifies existence of a tradeoff between rate and accuracy of convergence. For large values of α the sequence generated by Network Newton converges faster to the optimal solution of (4), while the accuracy of convergence is low since the penalty factor α is large. Smaller α implies smaller gap between the optimal solutions of (4) and (1), but the convergence rate of NN-K is slower.

Given that NN-K tries to approximate the Newton step, we expect to observe some form of superlinear convergence. The following lemma shows that convergence properties of NN-K are similar to those of Newton with constant stepsize. The difference is the appearance of a term that corresponds to the error of the Hessian inverse approximation.

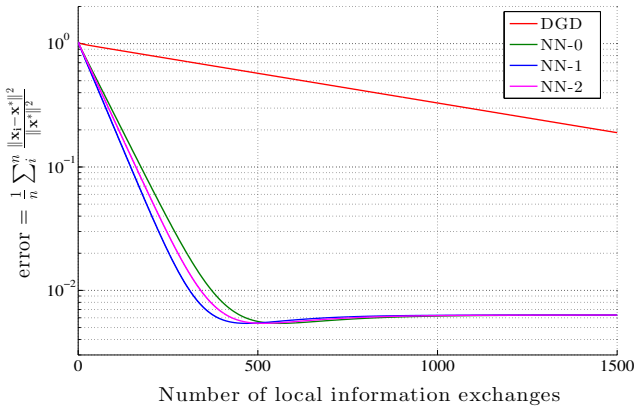

Fig. 1. Convergence of DGD, NN-0, NN-1, and NN-2. NN methods require less information exchanges than DGD.

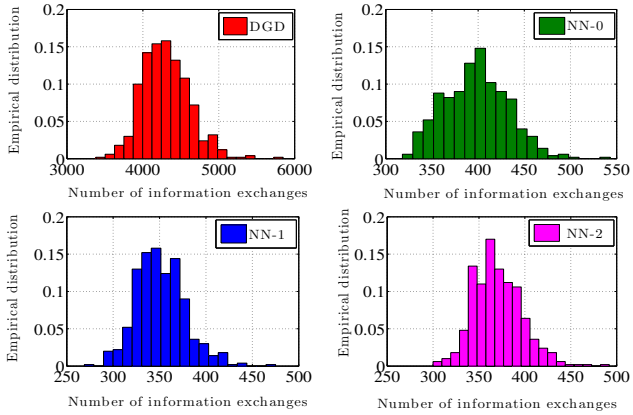

Fig. 2. Histograms of local information exchanges until achieving accuracy $e = 10^{-2}$. The required number of information exchanges for NN methods are significantly smaller than DGD.

Lemma 1 *Consider the Network Newton-K method as defined in* (9)-(14). *If Assumptions 1, 2, and 3 hold true, then*

$$\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\| \leq (1-\epsilon+\epsilon\rho^{K+1})\left[1+\Gamma_1(1-\zeta)^{(t-1)/4}\right]\left\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\right\| + \epsilon^2\Gamma_2\left\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\right\|^2, \tag{25}$$

where Γ_1 and Γ_2 are defined as

$$\Gamma_1 := \sqrt{\frac{\alpha\epsilon L(K+1)(2+\alpha M)}{(2(1-\Delta)+\alpha m)^3}}, \Gamma_2 := \frac{\alpha L(K+1)^2(2+\alpha M)}{2(2(1-\Delta)+\alpha m)^{\frac{5}{2}}}. \tag{26}$$

As per Lemma 1 the weighted gradient norm $\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\|$ is upper bounded by terms that are linear and quadratic on the weighted norm $\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$ associated with the previous iterate. This is akin to the gradient norm decrease of Newton's method. Further note that for all except the first few iterations $\Gamma_1(1-\zeta)^{(t-1)/4} \approx 0$. In that case the coefficient in the linear term reduces to $(1-\epsilon+\epsilon\rho^{K+1})$. This term can be reduced by making $\epsilon = 1$ and letting K grow, implying that there must be intervals in which the quadratic term dominates and NN-K exhibits quadratic convergence. We state this fact formally in the following theorem.

Theorem 2 *Consider the NN-K method as introduced in* (9)-(14) *and define the sequence $\eta_t := [(1-\epsilon+\epsilon\rho^{K+1})(1+\Gamma_1(1-\zeta)^{(t-1)/4})]$. Further, define t_0 as the first time that sequence η_t is smaller than 1, i.e. $t_0 := \text{argmin}_t\{t \mid \eta_t < 1\}$. If Assumptions 1, 2, and 3 hold true, then for all $t \geq t_0$ when the sequence $\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$ satisfies*

$$\frac{\sqrt{\eta_t}(1-\sqrt{\eta_t})}{\epsilon^2\Gamma_2} < \|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\| < \frac{1-\sqrt{\eta_t}}{\epsilon^2\Gamma_2}, \tag{27}$$

the rate of convergence is quadratic in the sense that

$$\|\mathbf{D}_t^{-1/2}\mathbf{g}_{t+1}\| \leq \frac{\epsilon^2\Gamma_2}{1-\sqrt{\eta_t}}\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|^2. \tag{28}$$

Theorem 2 shows that the sequence of weighted gradients $\|\mathbf{D}_{t-1}^{-1/2}\mathbf{g}_t\|$ generated by NN-K decays quadratically in the specified interval. That interval is not empty, since for $t \geq t_0$ we have $\sqrt{\eta_t} < 1$.

5. NUMERICAL ANALYSIS

We compare the number of information exchanges needed by DGD and different versions of NN to minimize a quadratic objective. In particular, for each agent i we consider a positive definite diagonal matrix $\mathbf{A}_i \in \mathbb{S}_p^{++}$ and a vector $\mathbf{b}_i \in \mathbb{R}^p$ to define the local object function $f_i(\mathbf{x}) := (1/2)\mathbf{x}^T\mathbf{A}_i\mathbf{x} + \mathbf{b}_i^T\mathbf{x}$. Therefore, the global cost function $f(\mathbf{x})$ is

$$f(\mathbf{x}) := \sum_{i=1}^n \frac{1}{2}\mathbf{x}^T\mathbf{A}_i\mathbf{x} + \mathbf{b}_i^T\mathbf{x}\,. \tag{29}$$

We pick $\mathbf{b}_i$ uniformly at random from the box $[0,1]^p$. To set a large condition number we generate matrices $\mathbf{A}_i$ as diagonal with elements a_{ii} where the first half of the components are uniformly drawn from the discrete set $\{1,\dots,10^{-\xi}\}$ and the second half of them from the set $\{1,\dots,10^{\xi}\}$. This choice of $\mathbf{A}_i$ yields a matrix $\sum_{i=1}^n \mathbf{A}_i$ that has eigenvalues in the interval $[n10^{-\xi}, n10^{\xi}]$. By setting larger ξ we can expect larger condition number for optimization problem (29). In all of our experiments we make the condition number constant $\xi = 2$, set the penalty coefficient to $\alpha = 0.01$, and for the NN step-size among powers of 10 we pick the one that performs better. The network size is $n = 100$ and the dimension of decision vectors is $p = 4$. We define relative error as the normalized average squared distance between decision vectors $\mathbf{x}_i$ and the optimal decision vector $\mathbf{x}^*$,

$$e := \frac{1}{n}\sum_{i=1}^n \frac{\|\mathbf{x}_i - \mathbf{x}^*\|^2}{\|\mathbf{x}^*\|^2}. \tag{30}$$

The graph of network in the experiments is d-regular, i.e., the degree of each node is d. The diagonal weights are set to $w_{ii} = 1/2 + 1/2(d+1)$ and the off diagonal weights to $w_{ij} = 1/2(d+1)$ when $j \in \mathcal{N}_i$. This implies that all diagonal weights satisfy $1/2 = \delta < w_{ii}$. We choose d uniformly at random from the integer values in the interval $[4, 10]$.

Fig. 1 illustrates the convergence path of DGD, NN-0, NN-1, and NN-2 by measuring the relative error e in (30) with respect to the total number of communication exchanges per node. For DGD this number coincides with the iteration index t. For NN-K the number of communication exchanges per node is $(K+1)t$. Fig. 1 shows that NN-0, NN-1, and NN-2 achieve accuracy $e = 10^{-2}$ after 3.7×10^2, 3.1×10^2, and 3.4×10^2 communications, respectively, while DGD after 1.5×10^3 information exchanges achieves accuracy $e = 1.9 \times 10^{-1}$. Fig. 2 shows the empirical distributions of the number of local communications between agents to achieve accuracy $e = 10^{-2}$ using 1000 realizations. According to Fig. 2 the average of local information exchanges for DGD is 4.3×10^3, while for NN-0, NN-1, and NN-2 the average of local communications are 4.0×10^2, 3.5×10^2, and 3.7×10^2, respectively. These numbers show the advantage of NN methods over DGD. Moreover, the optimal choice among different versions of NN-K is $K = 1$.

6. REFERENCES

[1] F. Bullo, J. Cortes, and S. Martinez, "Distributed control of robotic networks," *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," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 427–438, 2013.

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

[4] I. Schizas, A. Ribeiro, and G. Giannakis, "Consensus in ad hoc wsns with noisy links - part i: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, pp. 350–364, 2008.

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

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

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

[8] K. Tsianos, S. Lawlor, and M. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," *In: Proceedings of Allerton Conference on Communication, Control, and Computing*, 2012.

[9] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *IEEE Signal Processing Magazine*, vol. 31, pp. 32–43, 2014.

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

[11] D. Jakovetic, J. Xavier, and J. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, pp. 1131–1146, 2014.

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

[13] 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.

[14] Q. Ling and A. Ribeiro, "Decentralized linearized alternating direction method of multipliers," *Proc. Int. Conf. Acoustics Speech Signal Process.*, pp. 5447–5451, 2014.

[15] 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.

[16] J. Duchi, A. Agarwal, and M. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, pp. 592–606, 2012.

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

[18] M. Zargham, A. Ribeiro, A. Jadbabaie, and A. Ozdaglar, "Accelerated dual descent for network optimization," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 905–920, April 2014.

[19] M. Zargham, A. Ribeiro, and A. Jadbabaie, "Accelerated backpressure algorithm," *arXiv preprint arXiv*, 1302.1475, 2013.

[20] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed newton method for network utility maximization-i: Algorithm," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2162–2175, September 2013.

[21] M. Zargham and A. R. A. Jadbabaie, "Accelerated dual descent for constrained convex network flow optimization," *IEEE 52nd Annual Conference on Decision and Control (CDC)*, pp. 1037–1042, 2013.

[22] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network newton," 2014, available at http://www.seas.upenn.edu/∼aryanm/wiki/NN.pdf.