Newton-like method with diagonal correction for distributed optimization

Dragana Bajović [*]   Dušan Jakovetić [*]   Nataša Krejić [†]

Nataša Krklec Jerinkić [†]

August 15, 2015

Abstract

We consider distributed optimization problems where networked nodes cooperatively minimize the sum of their locally known convex costs. A popular class of methods to solve these problems are the distributed gradient methods, which are attractive due to their inexpensive iterations, but have a drawback of slow convergence rates. This motivates the incorporation of second-order information in the distributed methods, but this task is challenging: although the Hessians which arise in the algorithm design respect the sparsity of the network, their inverses are dense, hence rendering distributed implementations difficult. We overcome this challenge and propose a class of distributed Newton-like methods, which we refer to as Distributed Quasi Newton (DQN). The DQN family approximates the Hessian inverse by: 1) splitting the Hessian into its diagonal and off-diagonal part, 2) inverting the diagonal part, and 3) approximating the inverse of the off-diagonal part through a weighted linear function. The approximation is parameterized by the tuning variables which correspond to different splittings of the Hessian and by different weightings of the off-diagonal Hessian part. Specific choices of the tuning variables give rise to different variants of the proposed general DQN method – dubbed DQN-0, DQN-1 and DQN-2 – which mutually trade-off communication and computational costs for convergence. Simulations illustrate that the proposed DQN methods compare favorably with existing alternatives.

Key words: Distributed optimization, second order methods, Newton-like methods, linear convergence.

AMS subject classification. 90C25, 90C53, 65K05

[*]Biosense Institute, University of Novi Sad, Zorana Djindjića 3, 21000 Novi Sad, Serbia, email:{dbajovic@uns.ac.rs, djakovet@uns.ac.rs}

[†]Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Trg Dositeja Obradovića 4, 21000 Novi Sad, Serbia. e-mail: {natasak@uns.ac.rs, natasa.krklec@dmi.uns.ac.rs}.

1 Introduction

We consider a connected network with n nodes, each of which has access to a local cost function $f_i : \mathbb{R}^p \to \mathbb{R}$, $i = 1, \ldots, n$. The objective for all nodes is to minimize the aggregate cost function $f : \mathbb{R}^p \to \mathbb{R}$, defined by

$$f(y) = \sum_{i=1}^{n} f_i(y). \qquad (1)$$

Problems of this form arise in many emerging applications like big data analytics, e.g., [7], distributed inference in sensor networks [28, 13, 18, 5], and distributed control, [24].

Various methods for solving (1) in a distributed manner are available in the literature. A class of methods based on gradient descent at each node and exchange of information between neighboring nodes is particularly popular, see [25, 26, 27, 11, 12, 6, 33, 30, 17]. Assuming that the local costs f_i's are strongly convex and have Lipschitz continuous gradients and that a constant step size α is used, these methods converge linearly to a solution neighborhood. With such methods, step size α controls the tradeoff between the convergence speed towards a solution neighborhood and the distance of the limit point from the actual solution, larger α means faster convergence but larger distance from the solution in the limit; see, e.g., [12], [19]. Distributed first order (gradient) methods allow for a penalty interpretation, where the distributed method is interpreted as a (centralized) gradient method applied on a carefully constructed penalty reformulation of the original problem (1); see [12], [19] for details.

Given the existence of well developed theory and efficient implementations of higher order methods in centralized optimization in general, there is a clear need to investigate the possibilities of employing higher order methods in distributed optimization as well. More specifically, for additive cost structures (1) we study here, a further motivation for developing distributed higher order methods comes from their previous success when applied to similar problems in the context of centralized optimization. For example, additive cost (1) is typical in machine learning applications where second order methods play an important role, see, e.g., [2, 3, 4]. Another similar class of problems arise in stochastic optimization, where the objective function is given in the form of mathematical expectation. Again, second order methods are successfully applied in centralized optimization, [10, 14, 15, 22, 23].

There have been several papers on distributed Newton-type methods. A distributed second-order methods for network utility maximization and network flow optimization are developed in [31] and [34] but on problem formulations different from (1). Network Newton method [19] aims at solving (1) and presents a family of distributed (approximate) Newton methods. The class of Newtrok Newton method, refereed to as NN, is extensively analyzed in [20, 21]. The proposed methods are based on the penalty interpretation, [12, 19], of the distributed gradient method in [25], and they approximate the Newton step through an ℓ-th order Taylor approximation of the Hessian inverse, $\ell = 0, 1, \ldots$

This approximation gives rise to different variants of methods within the family, dubbed NN-ℓ. Different choices of ℓ exhibit inherent tradeoffs between the communication cost and the number of iterations until convergence, while NN-0, 1, and 2 are the most efficient in practice. The proposed methods converge linearly to a solution neighborhood, exhibit a kind of quadratic convergence phase, and show significantly better simulated performance when compared with the standard distributed gradient method in [25].

In this paper, we propose an alternative to methods in [19], i.e., a different family of distributed Newton-like methods for solving (1). We refer to the proposed family as Distributed Quasi Newton methods (DQN). The methods are designed to exploit the specific structure of the penalty reformulation [12, 19], as is done in [19], but with a different Hessian inverse approximation, for which the idea originates in [16]. Specifically, the Hessian matrix is approximated by its block diagonal part, while the remaining part of the Hessian is used to correct the right hand side of the quasi Newton equation. In such approximation one tries to preserve as much of the second order information as possible. The methods exhibit linear convergence to a solution neighbourhood under a set of standard assumptions for the functions f_i and the network architecture – each f_i is strongly convex and has Lipschitz continuous gradient, and the underlying network is connected. Simulation examples on (strongly convex) quadratic and logistic losses demonstrate that DQN compares favourably with NN proposed in [19].

With the DQN family of methods, the approximation of the Newton step is parameterized by diagonal matrix $\mathbb{L}_k$ in each iteration k and different choices of $\mathbb{L}_k$ give rise to different variants of DQN, which we refer to as DQN-0, 1, and 2. Specific choices of $\mathbb{L}_k$ are discussed in this paper. Different variants of DQN, based on different matrices $\mathbb{L}_k$ tradeoff the number of iterations and computational cost.

Let us further specify the relation between the proposed DQN family and NN methods in [19] as the NN methods are used as the benchmark in the work presented here. The Hessian approximation parameterized with weight matrix $\mathbb{L}_k$ and scalar $\theta \geq 0$ used in DQN family is in general very different from the Taylor expansion approximation employed in [19]. Actually, DQN can be considered a generalization of NN-0 and NN-1, where we recover NN-0 and NN-1 through certain specific choices of $\mathbb{L}_k$ and θ, as explained in Section 3. Further, with the proposed variants of DQN-0, 1, and 2, we utilize *diagonal matrices* $\mathbb{L}_k$ which results in computationally less expensive updates from the respective NN counterparts. Finally, the analysis here is very different from [19], and the major reason comes from the fact that the Hessian approximation with DQN is not a symmetric matrix in general. This fact also incurs the need for a safeguarding step with DQN in general, as detailed in Section 3. We also point out that results presented in [19] show that NN methods exhibit a quadratic convergence phase. It is likely that similar results can be shown for certain variants of DQN methods as well, but detailed analysis is left for future work.

Henceforth, in summary, the main purpose of this paper is to propose a class of distributed Newton-like methods which are inexpensive from both commu-

nication and computational points of view. The whole class is characterized by diagonal matrices $\mathbb{L}_k$'s and non-unit values of the splitting parameter θ (as opposed to [19] where $\theta = 1$), The convergence rate analysis is provided and their efficiency is demonstrated through simulation examples.

This paper is organized as follows. In Section 2 we give the problem statement and some preliminaries needed for the definition of the method and convergence analysis. Section 3 contains the description of the proposed class of Newton-like method and convergence results. Specific choices of the diagonal matrix that specifies the method completely are presented in Section 4. Some simulation results are presented in Section 5 while the conclusions are drawn in Section 6.

2   Preliminaries

Let us first give some preliminaries about the problem (1), its penalty interpretation in [12, 19], as well as the decentralized gradient descent algorithm in [25] that will be used later on.

The following assumption on the f_i's is imposed.

Assumption A1. The functions $f_i : \mathbb{R}^p \to \mathbb{R}$, $i = 1, \ldots, n$ are twice continuously differentiable, and there exist constants $0 < \mu \leq L < \infty$ such that for every $x \in \mathbb{R}^p$

$$\mu I \preceq \nabla^2 f_i(x) \preceq LI.$$

Here, I denotes the $p \times p$ identity matrix, and notation $M \preceq N$ means that the matrix $N - M$ is positive semi-definite.

This assumption implies that the functions f_i, $i = 1, \ldots, n$ are strongly convex with modulus $\mu > 0$,

$$f_i(z) \geq f_i(y) + \nabla f_i(y)^T (z - y) + \frac{\mu}{2} \|z - y\|^2, \ y, z \in \mathbb{R}^p, \tag{2}$$

and the gradients are Lipschitz continuous with the constant L i.e.

$$\|\nabla f_i(y) - \nabla f_i(z)\| \leq L \|y - z\|, \ y, z \in \mathbb{R}^p, \ i = 1, \ldots, n. \tag{3}$$

Assume that the network of nodes is an undirected network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of all edges, i.e., all pairs $\{i, j\}$ of nodes which can exchange information through a communication link.

Assumption A2. The network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected, undirected and simple (no self-loops nor multiple links).

Let us denote by O_i the set of nodes that are connected with the node i (open neighborhood of node i) and let $\bar{O}_i = O_i \bigcup \{i\}$ (closed neighborhood of node i). We associate with $\mathcal{G}$ a symmetric, (double) stochastic $n \times n$ matrix W. The elements of W are all nonnegative and rows (and columns) sum up to one. More precisely, we assume the following.

Assumption A3. The matrix $W = W^T \in \mathbb{R}^{n \times n}$ is stochastic with elements w_{ij} such that

$$w_{ij} > 0 \text{ if } \{i, j\} \in \mathcal{E}, \ w_{ij} = 0 \text{ if } \{i, j\} \notin \mathcal{E}, \ i \neq j, \text{ and } w_{ii} = 1 - \sum_{j \in O_i} w_{ij}$$

and there are constants w_{min} and w_{max} such that for $i = 1, \ldots, n$

$$0 < w_{min} \leq w_{ii} \leq w_{max} < 1.$$

Denote by $\lambda_1 \geq \ldots \geq \lambda_n$ the eigenvalues of W. Then it can be easily seen that $\lambda_1 = 1$. Furthermore, the null space of $I - W$ is spanned by $e := (1, \ldots, 1)$.

Following [12], [19], the auxiliary function $\Phi : \mathbb{R}^{np} \to \mathbb{R}$, and the corresponding penalty reformulation of (1) is introduced as follows. Let $x = (x_1, \ldots, x_n) \in \mathbb{R}^{np}$ with $x_i \in \mathbb{R}^p$, and denote by $\mathbb{Z} \in \mathbb{R}^{np \times np}$ the matrix obtained as the Kronecker product of W and the identity $I \in \mathbf{R}^{p \times p}, \mathbb{Z} = W \otimes I$.

The corresponding penalty reformulation of (1) is given by

$$\min_{x \in \mathbb{R}^{np}} \Phi(x) := \alpha \sum_{i=1}^{n} f_i(x_i) + \frac{1}{2} x^T (\mathbb{I} - \mathbb{Z}) x. \tag{4}$$

Applying the standard gradient method to (4) with the unit step size we get

$$x^{k+1} = x^k - \nabla \Phi(x^k), \ k = 0, 1, \ldots, \tag{5}$$

which, denoting the i-th $p \times 1$ block of x^k by x_i^k, and after rearranging terms, yields the Decentralized Gradient Descent (DGD) method [25]

$$x_i^{k+1} = \sum_{j \in \bar{O}_i} w_{ij} \, x_j^k - \alpha \nabla f_i(x_i^k), \ i = 1, \ldots, n. \tag{6}$$

Clearly, the penalty parameter α influences the relation between (4) and (1) – a smaller α means better agreement between the problems but also implies smaller steps in (6) and thus makes the convergence slower. It can be shown, [19] that if $\tilde{y} \in \mathbb{R}^p$ is the solution of (1) and $x^* := (\bar{y}_1, \ldots, \bar{y}_n) \in \mathbb{R}^{np}$ is the solution of (4) then, for all i,

$$\|\bar{y}_i - \tilde{y}\| = \mathcal{O}(\alpha). \tag{7}$$

The convergence of (6) towards x^* is linear, i.e., the following estimate holds [33, 12],

$$\Phi(x^k) - \Phi(x^*) \leq (1 - \xi)^k (\Phi(x^0) - \Phi(x^*)), \tag{8}$$

where $\xi \in (0, 1)$ is a constant depending on Φ, α and W.

The matrix and vector norms that will be used in the sequel are defined here. Let $\| \cdot \|$ denote an arbitrary vector norm on $\mathbb{R}^n$ and the corresponding matrix norm on $\mathbb{R}^{n \times n}$. For a matrix $\mathbb{M} \in \mathbb{R}^{np \times np}$ with blocks $M_{ij} \in \mathbb{R}^{p \times p}$, we define

$$\|\mathbb{M}\| := \max_{j=1,\ldots,n} \sum_{i=1}^{n} \|M_{ij}\|_2,$$

5

where $\| \cdot \|_2$ is the Euclidean norm. For a vector $x \in \mathbb{R}^{np}$ with blocks $x_i \in \mathbb{R}^p$ the norm is defined as

$$\|x\| := \sum_{i=1}^{n} \|x_i\|_2.$$

3 Distributed Quasi Newton method

In this section we introduce a class of Quasi Newton methods for solving (4). The general Distributed Quasi Newton (DQN) method is proposed in Subsection 3.1. The method is characterized by a generic diagonal matrix $\mathbb{L}_k$. Global linear convergence rate for the class is established in Subsection 3.2, while local linear convergence rate with the full step size $\varepsilon = 1$ is analyzed in Subsection 3.3. Specific variants DQN-0, 1, and 2, which correspond to the specific choices of $\mathbb{L}_k$, are studied in Section 4. As we will see, algorithm DQN has certain tuning parameters, including the step size ε. Discussion on the tuning parameters choice is relegated to Section 4.

3.1 The proposed general DQN method

The problem we consider from now on is (4), where we recall $\mathbb{Z} = W \otimes I$ and W satisfies assumption A3.

The problem under consideration has a specific structure as the Hessian is sparse if the underlying network is sparse. However its inverse is dense. Furthermore, the matrix inversion (i.e. linear system solving) is not suitable for decentralized computation. One possibility of exploiting the structure of $\nabla^2 \Phi(x)$ in a distributed environment is presented in [19] where the Newton step is approximated through a number of inner iterations. We present here a different possibility. Namely, we keep the diagonal part of $\nabla^2 \Phi(x)$ as the Hessian approximation but at the same time use the non-diagonal part of $\nabla^2 \Phi(x)$ to correct the right hand side vector in the (quasi)-Newton equation. Let us define the splitting

$$W_d = diag(W) \quad \text{and} \quad W_u = W - W_d,$$

and $\mathbb{Z} = \mathbb{Z}_d + \mathbb{Z}_u$ with

$$\mathbb{Z}_d = W_d \otimes I = diag(\mathbb{Z}) \quad \text{and} \quad \mathbb{Z}_u = W_u \otimes I.$$

Here, $diag(W)$ denotes the diagonal matrix with the same diagonal as the matrix W. Hence, matrix $\mathbb{Z}_d$ is a $np \times np$ (block) diagonal matrix whose i-th $p \times p$ block is the scalar matrix $w_{ii}I$, $\mathbb{Z}_u$ is a $np \times np$ block (symmetric) matrix such that (i, j)-th $p \times p$ off-diagonal blocks are again scalar matrices $w_{ij}I$, while the diagonal blocks are all equal to zero.

Clearly, the gradient is

$$\nabla \Phi(x) = \alpha \nabla F(x) + (\mathbb{I} - \mathbb{Z})x,$$

where $\mathbb{I}$ denotes the $np \times np$ identity matrix and

$$F(x) = \sum_{i=1}^{n} f_i(x_i), \quad \nabla F(x) = (\nabla f_1(x_1), \ldots, \nabla f_n(x_n)),$$

while the Hessian is
$$\nabla^2 \Phi(x) = \alpha \nabla^2 F(x) + \mathbb{I} - \mathbb{Z}$$

where $\nabla^2 F(x)$ is the block diagonal matrix with the ith diagonal block $\nabla^2 f_i(x_i)$.

The general Distributed Quasi Newton, DQN, algorithm is presented below. Denote by k the iteration counter, $k = 0, 1, \ldots$, and let $x^k = (x_1^k, \ldots, x_n^k) \in \mathbb{R}^{np}$ be the estimate of x^* at iteration k. Consider the following splitting of the Hessian
$$\nabla^2 \Phi(x^k) = \mathbb{A}_k - \mathbb{G}, \tag{9}$$

with
$$\mathbb{A}_k = \alpha \nabla^2 F(x^k) + (1+\theta)(\mathbb{I} - \mathbb{Z}_d) \tag{10}$$

and
$$\mathbb{G} = \mathbb{Z}_u + \theta(\mathbb{I} - \mathbb{Z}_d)$$

for some $\theta \geq 0$. Hence, $\mathbb{G}$ is a $np \times np$ block (symmetric) matrix whose i-th $p \times p$ diagonal block equals $g_{ii} I$, with $g_{ii} := \theta (1 - w_{ii})$, while the (i,j)-th $p \times p$ off-diagonal block equals $g_{ij} I$, with $g_{ij} := w_{ij}$. Also, notice that $\mathbb{A}_k$ is block diagonal with the ith diagonal block

$$A_i^k = \alpha \nabla^2 f_i(x_i^k) + (1+\theta)(1 - w_{ii})I.$$

Let $\mathbb{L}_k \in \mathbb{R}^{np \times np}$ be a block-diagonal, symmetric matrix with the i-th $p \times p$ block given by the symmetric matrix Λ_i^k. Given that

$$\nabla^2 \Phi(x^k) = \mathbb{A}_k - \mathbb{G} = \mathbb{A}_K(\mathbb{I} - \mathbb{A}_k^{-1}\mathbb{G}),$$

let us consider the approximation

$$(\nabla^2 \Phi(x^k))^{-1} = (\mathbb{I} - \mathbb{A}_k^{-1}\mathbb{G})^{-1}\mathbb{A}_k^{-1} \approx (\mathbb{I} - \mathbb{L}_k\mathbb{G})\mathbb{A}_k^{-1}. \tag{11}$$

In other words, we approximate the dense part of the inverse Hessian $(\mathbb{I} - \mathbb{A}_k^{-1}\mathbb{G})^{-1}$ by a matrix function linear in $\mathbb{G}$, equal to $\mathbb{I} - \mathbb{L}_k\mathbb{G}$. The Hessian inverse approximation in (11) is amenable for distributed implementation, as the resulting matrix $(\mathbb{I} - \mathbb{L}_k\mathbb{G})\mathbb{A}_k^{-1}$ respects the sparsity pattern of the network, in the sense that its (i,j)-th $p \times p$ block equals zero if $(i,j) \notin E$, for each pair $i \neq j$.

Setting $\mathbb{L}_k = 0$ corresponds to the zeroth order Taylor approximation of $(\mathbb{I} - \mathbb{A}_k^{-1}\mathbb{G})^{-1}$, while $\mathbb{L}_k = -\mathbb{A}_k^{-1}$ corresponds to the first order Taylor approximation of $(\mathbb{I} - \mathbb{A}_k^{-1}\mathbb{G})^{-1}$.[1] In general, allowing for different choices of $\mathbb{L}_k$ gives rise to

---

[1] The Taylor expansion $(\mathbb{I} - \mathbb{A}_k^{-1}\mathbb{G})^{-1} = \sum_{j=0}^{+\infty}(\mathbb{A}_k^{-1}\mathbb{G})^j$ is well defined, as the spectral radius of matrix $\mathbb{A}_k^{-1}\mathbb{G}$ is strictly less than one for the set of parameters that we study; see ahead Theorem 3.4 and (25).

more general approximations than the Taylor ones. We will be interested in *diagonal* $\mathbb{L}_k$'s, which is in a sense less expensive approximations than the first order Taylor approximation. In this paper, we do not claim nor seek optimality of approximation (11). Instead, our goal is to show that approximation (11) and inexpensive (in terms of computations and communications), diagonal matrix choices of $\mathbb{L}_k$ – which we design here – yield efficient distributed Newton-like algorithms.

For such approximation the search direction is defined as

$$s^k = -(\mathbb{I} - \mathbb{L}_k \mathbb{G}) \mathbb{A}_k^{-1} \nabla \Phi(x^k). \tag{12}$$

Then, following typical quasi-Newton scheme, the next iteration is defined by

$$x^{k+1} = x^k + \varepsilon s^k, \tag{13}$$

for some step size ε.

Clearly, the choice of $\mathbb{L}_k$ is crucial in the approximation of $(\nabla^2 \Phi(x^k))^{-1}$. The following general algorithm assumes only that $\mathbb{L}_k$ is diagonal and bounded. Specific choices of $\mathbb{L}_k$ will be discussed in Section 4. Actually, all the proposed variants DQN-0, 1, and 2 utilize *diagonal* matrices $\mathbb{L}_k$. Parameter θ affects splitting (9), approximation (11) and the search direction (12). For this moment we are assuming only that θ is nonnegative and further details are presented later on.

In summary, the proposed distributed algorithm is as follows.

Algorithm DQN

Given $x^0 \in \mathbb{R}^{np}, \alpha, \varepsilon, \rho > 0, \theta \geq 0$. Set $k = 0$.

Step 1. Chose a diagonal matrix $\mathbb{L}_k \in \mathbb{R}^{np \times np}$ such that

$$\|\mathbb{L}_k\| \leq \rho.$$

Step 2. Set

$$s^k = -(\mathbb{I} - \mathbb{L}_k \mathbb{G}) \mathbb{A}_k^{-1} \nabla \Phi(x^k).$$

Step 3. Set

$$x^{k+1} = x^k + \varepsilon s^k, \ k = k + 1.$$

For the sake of clarity, the proposed algorithm, from the perspective of each node i in the network, is presented in Algorithm 1.

Algorithm 1

At each node i, require $\alpha, \rho, \varepsilon > 0, \theta \geq 0$.

1 Initialization: Each node i sets $k = 0$ and $x_i^0 \in \mathbb{R}^p$.

2 Each node i transmits x_i^k to all its neighbors $j \in O_i$ and receives x_j^k from all $j \in O_i$.

3 Each node i calculates

$$d_i^k = \left(A_i^k\right)^{-1} \left[\alpha \nabla f_i(x_i^k) + \sum_{j \in O_i} w_{ij}\left(x_i^k - x_j^k\right)\right].$$

4 Each node i transmits d_i^k to all its neighbors $j \in O_i$ and receives d_j^k from all $j \in O_i$.

5 Each node i chooses a diagonal $p \times p$ matrix Λ_i^k, such that $\|\Lambda_i^k\| \leq \rho$.

6 Each node i calculates:

$$s_i^k = -d_i^k + \Lambda_i^k \sum_{j \in \bar{O}_i} G_{ij}\, d_j^k.$$

7 Each node i updates its solution estimate as:

$$x_i^{k+1} = x_i^k + \varepsilon\, s_i^k.$$

8 Set $k = k + 1$ and go to step 3.

Calculation of Λ_i^k in step 6 will be specified in the next Section, and, for certain algorithm variants, will involve an additional inter-neighbor communication of a p-dimensional vector. Likewise, choices of tuning parameters $\alpha, \varepsilon, \rho, \theta$ are discussed throughout the remaining of this section and Section 4.

Remark. It is useful to compare (11) with the Hessian approximation in NN methods. Setting $\theta = 1$ and $\mathbb{L}_k = 0$ recovers NN-0, $\theta = 1$ and $\mathbb{L}_k = -\mathbb{A}_k^{-1}$ recovers NN-1, while NN-2 can not be recovered in a similar fashion. Hence, DQN in a sense generalizes NN-0 and NN-1. Next, observe that the approximation matrix in (11) is not symmetric in general. This induces the need for a safeguarding step in the algorithm which will be shown later on, and the resulting method requires an analysis different from [19].

3.2   Global linear convergence rate

In this subsection, the global linear convergence rate of Algorithm DQN is established. The convergence analysis consists of two parts. First, we demonstrate that s^k is a descent direction. Then we determine a suitable interval for the step size ε that ensures linear convergence of the iterative sequence.

The following Gershgorin type theorem for block matrices is needed for the first part of convergence analysis.

Theorem 3.1. *[9] For any $\mathbb{C} \in \mathbb{R}^{np \times np}$ partitioned into blocks C_{ij} of size p, each eigenvalue μ of $\mathbb{C}$ satisfies*

$$\frac{1}{\|(C_{ii} - \mu I)^{-1}\|} \leq \sum_{i \neq j} \|C_{ij}\| \tag{14}$$

for at least one $i \in \{1, \ldots, n\}$.

Using the above theorem we can prove the following lower bound for all eigenvalues of a symmetric block matrix.

Corollary 3.1. *Let* $\mathbb{C} \in \mathbb{R}^{np \times np}$ *be a symmetric matrix partitioned into blocks* C_{ij} *of size p. Then each eigenvalue* μ *of* $\mathbb{C}$ *satisfies*

$$\mu \geq \min_{i=1,\dots,n} \left\{ \lambda_{\min}(C_{ii}) - \sum_{j \neq i} \|C_{ij}\| \right\},$$

where $\lambda_{\min}(C_{ii})$ *is the smallest eigenvalue of* C_{ii}.

Proof. Given that $\mathbb{C}$ is symmetric, all its eigenvalues are real. Also, C_{ii} is symmetric and has only real eigenvalues. Now, fix one eigenvalue μ of the matrix $\mathbb{C}$. By Theorem 3.1, there exists $i \in \{1, \dots, n\}$, such that (14) holds. Next, we have

$$\|(C_{ii} - \mu I)^{-1}\| = \frac{1}{\min_{j=1,\dots,p} |\lambda_j(C_{ii}) - \mu|},$$

where $\lambda_j(C_{ii})$ is the j-th eigenvalue of C_{ii}. Thus

$$\min_{j=1,\dots,p} |\lambda_j(C_{ii}) - \mu| \leq \sum_{j \neq i} \|C_{ij}\|.$$

We have just concluded that, for any eigenvalue μ of $\mathbb{C}$ there exists $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, p\}$ such that μ lies in the interval

$$[\lambda_j(C_{ii}) - \sum_{i \neq l} \|C_{il}\|, \lambda_j(C_{ii}) + \sum_{i \neq l} \|C_{il}\|].$$

Hence, for each μ for which (14) holds for some fixed i, we have

$$\mu \geq \lambda_{\min}(C_{ii}) - \sum_{l \neq i} \|C_{il}\|$$

and the statement follows. $\square$

We are now ready to prove that the search direction (12) is descent.

Theorem 3.2. *Suppose that A1-A3 hold. Let*

$$0 \leq \rho \leq \frac{\alpha \mu + (1+\theta)(1 - w_{max})}{(1 - w_{min})(1+\theta)} \left(\frac{1}{\alpha L + (1+\theta)(1 - w_{min})} - \delta \right) \quad (15)$$

for some $\delta \in (0, 1/(\alpha L + (1+\theta)(1 - w_{min})))$. *Then* s^k *defined by (12) is a descent direction and satisfies*

$$\nabla^T \Phi(x^k) s^k \leq -\delta \|\nabla \Phi(x^k)\|^2.$$

Proof. Let us first show that s^k is descent search direction. As

$$\nabla^T \Phi(x^k) s^k = -\nabla^T \Phi(x^k)(\mathbb{I} - \mathbb{L}_k \mathbb{G})\mathbb{A}_k^{-1} \nabla \Phi(x^k),$$

s^k is descent if $v^T(\mathbb{I} - \mathbb{L}_k \mathbb{G})\mathbb{A}_k^{-1} v < 0$ for arbitrary $v \in \mathbb{R}^{np \times np}$. Given that $(\mathbb{I} - \mathbb{L}_k \mathbb{G})\mathbb{A}_k^{-1}$ is not symmetric in general, we know the above is true if and only if the matrix

$$\mathbb{C}^k = \frac{1}{2}((\mathbb{I} - \mathbb{L}_k \mathbb{G})\mathbb{A}_k^{-1} + \mathbb{A}_k^{-1}(\mathbb{I} - \mathbb{G}\mathbb{L}_k))$$

is positive definite. $\mathbb{C}^k$ is symmetric and thus it is positive definite if all of its eigenvalues are positive. The matrix $\mathbb{C}^k$ is partitioned in the blocks

$$C_{ii}^k = (A_i^k)^{-1} - \frac{1}{2}\theta(1 - w_{ii})(\Lambda_i^k (A_i^k)^{-1} + (A_i^k)^{-1}\Lambda_i^k), \quad i = 1, \ldots, n,$$

$$C_{ij}^k = -\frac{1}{2}w_{ij}(\Lambda_i^k (A_j^k)^{-1} + (A_i^k)^{-1}\Lambda_j^k), \quad i \neq j.$$

Corollary 3.1 implies that

$$\lambda_{min}(\mathbb{C}^k) \geq \min_{i=1,\ldots,n}(\lambda_{min}(C_{ii}^k) - \sum_{j \neq i}\|C_{ij}^k\|_2)$$

The definition of $\mathbb{A}_k$ implies

$$(\alpha\mu + (1 + \theta)(1 - w_{max}))I \preceq A_i^k \preceq (\alpha L + (1 + \theta)(1 - w_{min}))I$$

and therefore, for every $i = 1, \ldots, n$

$$\|(A_i^k)^{-1}\|_2 \leq \frac{1}{\alpha\mu + (1 + \theta)(1 - w_{max})}.$$

Moreover, it follows

$$\lambda_{min}(C_{ii}^k) \geq \frac{1}{\alpha L + (1 + \theta)(1 - w_{min})} - \frac{\theta(1 - w_{ii})\rho}{\alpha\mu + (1 + \theta)(1 - w_{max})}$$

and

$$\|C_{ij}^k\|_2 \leq \frac{w_{ij}\rho}{\alpha\mu + (1 + \theta)(1 - w_{max})}.$$

Now,

$$
\begin{aligned}
\lambda_{min}(C^k) &\geq \min_{i=1,\ldots,n}\left(\frac{1}{\alpha L + (1 + \theta)(1 - w_{min})} - \frac{\theta(1 - w_{ii})\rho}{\alpha\mu + (1 + \theta)(1 - w_{max})}\right.\\
&\quad - \sum_{j \in O_i} w_{ij}\frac{\rho}{\alpha\mu + (1 + \theta)(1 - w_{max})}\bigg)\\
&= \min_{i=1,\ldots,n}\left(\frac{1}{\alpha L + (1 + \theta)(1 - w_{min})} - \frac{\rho(1 - w_{ii})(1 + \theta)}{\alpha\mu + (1 + \theta)(1 - w_{max})}\right)\\
&\geq \frac{1}{\alpha L + (1 + \theta)(1 - w_{min})} - \rho\frac{(1 - w_{min})(1 + \theta)}{\alpha\mu + (1 + \theta)(1 - w_{max})}\\
&\geq \delta. \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (16)
\end{aligned}
$$

11

Since $\delta > 0$ we conclude that $\mathbb{C}^k$ is positive definite. Moreover, $v^T \mathbb{C}^k v = v^T (\mathbb{I} - \mathbb{L}_k \mathbb{G}) \mathbb{A}_k^{-1} v$, for any $v \in \mathbb{R}^{np \times np}$ and

$$
\begin{aligned}
\nabla^T \Phi(x^k) s^k &= -\nabla^T \Phi(x^k)(\mathbb{I} - \mathbb{L}_k \mathbb{G}) \mathbb{A}_k^{-1} \nabla \Phi(x^k) \\
&= -\nabla^T \Phi(x^k) C^k \nabla \Phi(x^k) \\
&\leq -\delta \|\nabla \Phi(x^k)\|^2.
\end{aligned}
\tag{17}
$$

□

The next lemma corresponds to the standard property of descent direction methods that establish the relationship between the search vector and the gradient.

Lemma 3.1. *Suppose that A1-A3 hold. Then*

$$
\|s^k\| \leq \beta \|\nabla \Phi(x^k)\|,
$$

where

$$
\beta = \frac{1 + \rho(1 + \theta)(1 - w_{min})}{\alpha \mu + (1 + \theta)(1 - w_{max})}.
\tag{18}
$$

Proof. The matrix $\mathbb{A}_k$ is block diagonal and therefore

$$
\|\mathbb{A}_k^{-1}\| = \max_{j=1,\dots,n} \|(A_j^k)^{-1}\|_2 \leq \frac{1}{\alpha \mu + (1 + \theta)(1 - w_{max})}.
\tag{19}
$$

Furthermore,

$$
\|\mathbb{G}\| = \max_{j=1,\dots,n} \left(\theta(1 - w_{jj}) + \sum_{i \in O_j} w_{ij} \right) = \max_{j=1,\dots,n} (1 + \theta)(1 - w_{jj}) \leq (1 + \theta)(1 - w_{min}).
\tag{20}
$$

So

$$
\begin{aligned}
\|s^k\| &\leq \|(\mathbb{I} - \mathbb{L}_k \mathbb{G}) \mathbb{A}_k^{-1}\| \|\nabla \Phi(x^k)\| \\
&\leq (1 + \|\mathbb{L}_k\| \|\mathbb{G}\|) \|\mathbb{A}_k^{-1}\| \|\nabla \Phi(x^k)\| \\
&\leq \frac{1 + \rho(1 + \theta)(1 - w_{min})}{\alpha \mu + (1 + \theta)(1 - w_{max})} \|\nabla \Phi(x^k)\|.
\end{aligned}
$$

□

Let us now show that there exists a step size $\varepsilon > 0$ such that the sequence $\{x^k\}$ generated by Algorithm DQN converges to the solution of (4). Notice that (4) has a unique solution, say x^*. Assumption A1 implies that $\nabla \Phi(x)$ is Lipschitz continuous as well, i.e., with $\tilde{L} := \alpha L + 2(1 - w_{min})$, there holds

$$
\|\nabla \Phi(x) - \nabla \Phi(y)\| \leq \tilde{L} \|x - y\|, \ x, y \in \mathbb{R}^{np}.
\tag{21}
$$

Furthermore,

$$
\frac{\tilde{\mu}}{2} \|x - x^*\| \leq \Phi(x) - \Phi(x^*) \leq \frac{1}{\tilde{\mu}} \|\nabla \Phi(x)\|^2
\tag{22}
$$

for $\tilde{\mu} = \alpha \mu$ and all $x \in \mathbb{R}^{np}$. The main convergence statement is given below.

12

Theorem 3.3. *Assume that the conditions of Theorem 3.2 are satisfied. Define*

$$\varepsilon = \frac{\delta}{\beta^2 \tilde{L}}$$

with β given by (18). Then Algorithm DQN generates a sequence $\{x^k\}$ such that

$$\lim_{k \to \infty} x^k = x^*$$

and the convergence is at least linear with

$$\Phi(x^{k+1}) - \Phi(x^*) \le \left(1 - \frac{\delta^2 \tilde{\mu}}{2 \tilde{L} \beta^2}\right) \left(\Phi(x^k) - \Phi(x^*)\right), \quad k = 0, 1, \dots.$$

Proof. The Mean Value Theorem, Lipschitz property of $\nabla \Phi$, Theorem 3.2 and Lemma 3.1 yield

$$
\begin{aligned}
\Phi(x^{k+1}) - \Phi(x^*) &= \Phi(x^k + \varepsilon s^k) - \Phi(x^*) \\
&= \Phi(x^k) + \int_0^1 \nabla^T \Phi(x^k + t\varepsilon s^k)\varepsilon s^k dt - \Phi(x^*) \pm \varepsilon \nabla^T \Phi(x^k) s^k \\
&\le \Phi(x^k) - \Phi(x^*) + \varepsilon \int_0^1 \|\nabla^T \Phi(x^k + t\varepsilon s^k) - \nabla^T \Phi(x^k)\| \|s^k\| dt \\
&\quad + \varepsilon \nabla^T \Phi(x^k) s^k \\
&\le \Phi(x^k) - \Phi(x^*) + \varepsilon \int_0^1 \tilde{L} t \varepsilon \|s^k\|^2 dt + \varepsilon \nabla^T \Phi(x^k) s^k \\
&= \Phi(x^k) - \Phi(x^*) + \frac{1}{2}\varepsilon^2 \tilde{L} \|s^k\|^2 + \varepsilon \nabla^T \Phi(x^k) s^k \\
&\le \Phi(x^k) - \Phi(x^*) + \beta^2 \frac{\tilde{L}}{2}\varepsilon^2 \|\nabla \Phi(x^k)\|^2 - \varepsilon\delta \|\nabla \Phi(x^k)\|^2 \\
&= \Phi(x^k) - \Phi(x^*) + \left(\frac{\beta^2 \tilde{L}}{2}\varepsilon^2 - \varepsilon\delta\right)\|\nabla^2 \Phi(x^k)\|^2. \quad (23)
\end{aligned}
$$

Define

$$\phi(\varepsilon) = \frac{\beta^2 \tilde{L}}{2}\varepsilon^2 - \varepsilon\delta.$$

Then $\phi(0) = 0$, $\phi'(\varepsilon) = \tilde{L}\varepsilon\beta^2 - \delta$ and $\phi''(\varepsilon) > 0$. Thus, the minimizer of ϕ is $\varepsilon^* = \delta/(\beta^2 \tilde{L})$ and

$$\phi(\varepsilon^*) = -\frac{\delta^2}{2\beta^2 \tilde{L}}. \quad (24)$$

Now, (23) and (24) give

$$\Phi(x^{k+1}) - \Phi(x^*) \le \Phi(x^k) - \Phi(x^*) - \frac{\delta^2}{2\beta^2 \tilde{L}}\|\nabla \Phi(x^k)\|^2.$$

13

From (22), we also have

$$\Phi(x^k) - \Phi(x^*) \leq \frac{1}{\tilde{\mu}}\|\nabla\Phi(x^k)\|^2$$

and

$$-\frac{\delta^2}{2\beta^2\tilde{L}}\|\nabla\Phi(x^k)\|^2 \leq -\left(\Phi(x^k) - \Phi(x^*)\right)\frac{\delta^2\tilde{\mu}}{2\beta^2\tilde{L}},$$

so

$$\Phi(x^{k+1}) - \Phi(x^*) \leq \left(1 - \frac{\delta^2\tilde{\mu}}{2\beta^2\tilde{L}}\right)\left(\Phi(x^k) - \Phi(x^*)\right).$$

Given that $\tilde{\mu} = \alpha\mu \leq \alpha L < \tilde{L}$, we have $\tilde{\mu}/\tilde{L} < 1$. Moreover,

$$
\begin{aligned}
\delta \quad &< \quad \frac{1}{\alpha L + (1+\theta)(1-w_{min})} \leq \frac{1}{\alpha\mu + (1+\theta)(1-w_{max})} \\
&\leq \quad \frac{1 + \rho(1+\theta)(1-w_{min})}{\alpha\mu + (1+\theta)(1-w_{max})} = \beta
\end{aligned}
$$

and

$$\xi := 1 - \frac{\delta^2\tilde{\mu}}{2\beta^2\tilde{L}} \in (0,1).$$

We conclude with

$$\Phi(x^{k+1}) - \Phi(x^*) \leq \xi\left(\Phi(x^k) - \Phi(x^*)\right)$$

and

$$\lim_{k\to\infty}\Phi(x^k) = \Phi(x^*).$$

As $\Phi \in C^2(\mathbb{R}^n)$, the above limit also implies

$$\lim_{k\to\infty} x^k = x^*.$$

$\square$

The proof of the above theorem clearly shows that for any $\varepsilon \in (0, \delta/(\beta^2\tilde{L})]$ algorithm DQN converges. However, taking ε as large as possible implies larger steps and thus faster convergence.

3.3 Local linear convergence

We have proved global linear convergence for the specific step length ε given in Theorem 3.3. However, local linear convergence can be obtained for the full step by imposing a stronger condition on the diagonal matrix $\mathbb{L}_k$, using the theory developed for Inexact Newton methods [8]. The step s^k can be considered as an Inexact Newton step and we are able to estimate the residual in Newton equation as follows.

Theorem 3.4. *Suppose that A1-A3 hold. Let x^k be such that $\nabla\Phi(x^k) \neq 0$. Assume that s^k is generated in Step 2 of Algorithm DQN with*

$$0 \leq \rho < \frac{\alpha\mu}{(1+\theta)(1 - w_{min})(\alpha L + 2(1+\theta)(1 - w_{min}))}.$$

Then there exists $t \in (0,1)$ such that

$$\|\nabla\Phi(x^k) + \nabla^2\Phi(x^k)s^k\| < t\|\nabla\Phi(x^k)\|.$$

Proof. First, notice that the interval for ρ is well defined. The definition of the search direction (12) and the splitting of the Hessian (9) yield

$$\nabla\Phi(x^k) + \nabla^2\Phi(x^k)s^k = (\mathbb{G}\mathbb{A}_k^{-1} + \mathbb{A}_k\mathbb{L}_k\mathbb{G}\mathbb{A}_k^{-1} - \mathbb{G}\mathbb{L}_k\mathbb{G}\mathbb{A}_k^{-1})\nabla\Phi(x^k) := \mathbb{Q}_k\nabla\Phi(x^k).$$

Therefore,

$$\|\mathbb{Q}_k\| \leq \|\mathbb{G}\mathbb{A}_k^{-1}\| + \|\mathbb{G}\mathbb{A}_k^{-1}\|\|\mathbb{L}_k\|\|\mathbb{G}\| + \|\mathbb{G}\mathbb{A}_k^{-1}\|\|\mathbb{L}_k\|\|\mathbb{A}_k\|.$$

Moreover,

$$
\begin{aligned}
\|\mathbb{G}\mathbb{A}_k^{-1}\| &= \max_j(\theta(1 - w_{jj})\|(A_j^k)^{-1}\| + \sum_{i \in O_j} w_{ij}\|(A_j^k)^{-1}\|) \\
&\leq \max_j \frac{\theta(1 - w_{jj}) + 1 - w_{jj}}{\alpha\mu + (1+\theta)(1 - w_{jj})} \\
&= \frac{(1+\theta)(1 - w_{min})}{\alpha\mu + (1+\theta)(1 - w_{min})} =: \gamma.
\end{aligned}
\tag{25}
$$

and there holds $\|A_k\| \leq \alpha L + (1+\theta)(1 - w_{min})$. Furthermore, (19), (20) and $\|\mathbb{L}_k\| \leq \rho$ imply

$$
\begin{aligned}
\|\mathbb{Q}_k\| &\leq \gamma + \gamma\rho(1+\theta)(1 - w_{min}) + \gamma\rho(\alpha L + (1+\theta)(1 - w_{min})) \\
&= \gamma + \rho\gamma(\alpha L + 2(1+\theta)(1 - w_{min})) \\
&< \gamma + 1 - \gamma = 1.
\end{aligned}
$$

Thus, the statements is true with

$$t = \gamma + \rho\gamma(\alpha L + 2(1+\theta)(1 - w_{min})). \tag{26}$$

□

For the sake of completeness we list here the conditions for local convergence of Inexact Newton methods.

Theorem 3.5. *[8] Assume that A1 holds and that s^k satisfies the inequality*

$$\|\nabla\Phi(x^k) + \nabla^2\Phi(x^k)s^k\| < t\|\nabla\Phi(x^k)\|, \quad k = 0, 1, \ldots$$

for some $t < 1$. Furthermore, assume that $x^{k+1} = x^k + s^k$, $k = 0, 1, \ldots$. Then there exists $\eta > 0$ such that for all $\|x^0 - x^*\| \leq \eta$, the sequence $\{x^k\}$ converges to x^*. The convergence is linear,

$$\|x^{k+1} - x^*\|_* \leq t\|x^k - x^*\|_*, k = 0, 1, \ldots,$$

where $\|y\|_* = \|\nabla^2\Phi(x^*)y\|$.

15

The two previous theorems imply the following Corollary.

Corollary 3.2. *Assume that the conditions of Theorem 3.4 hold. Then there exists $\eta > 0$ such that for every x^0 satisfying $\|x^0 - x^*\| \le \eta$, the sequence $\{x^k\}$ generated by Algorithm DQN and $\varepsilon = 1$ converges linearly to x^* and*

$$\|\nabla^2\Phi(x^*)(x^{k+1} - x^*)\| \le t\|\nabla^2\Phi(x^*)(x^k - x^*)\|, \quad k = 0, 1, \ldots$$

holds with $t \in (0, 1)$.

For (strongly convex) quadratic functions f_i, $i = 1, \ldots, n$ we can also claim global linear convergence as follows.

Theorem 3.6. *Assume that all loss functions f_i are strongly convex quadratic and that the conditions of Theorem 3.4 are satisfied. Let $\{x^k\}$ be a sequence generated by Algorithm DQN with $\varepsilon = 1$. Then $\lim_{k\to\infty} x^k = x^*$ and*

$$\|x^{k+1} - x^*\|_* \le t\,\|x^k - x^*\|_*, \quad k = 0, 1, \ldots$$

for t defined in Theorem 3.4.

Proof. Given that the penalty term in (4) is convex quadratic, if all local cost functions f_i are strongly convex quadratic, then the objective function Φ is also strongly convex quadratic, i.e., it can be written as

$$\Phi(x) = \frac{1}{2}(x - x^*)^T\mathbb{B}(x - x^*), \tag{27}$$

for some fixed, symmetric positive definite matrix $\mathbb{B} \in \mathbb{R}^{np \times np}$. Recall that x^* is the global minimizer of Φ. Then

$$\nabla\Phi(x) = \mathbb{B}(x - x^*) \text{ and } \nabla^2\Phi(x) = \mathbb{B}.$$

Starting from

$$s^k = -(\nabla^2\Phi(x^k))^{-1}\nabla\Phi(x^k) + e^k,$$

we get

$$
\begin{aligned}
\|\nabla^2\Phi(x^k)e^k\| &= \|\nabla^2\Phi(x^k)\left(s^k + (\nabla^2\Phi(x^k))^{-1}\nabla\Phi(x^k)\right)\| \\
&= \|\nabla\Phi(x^k) + \nabla^2\Phi(x^k)s^k\| < t\|\nabla\Phi(x^k)\|
\end{aligned}
$$

by Theorem 3.4. Next,

$$
\begin{aligned}
x^{k+1} &= x^k + s^k = x^k - (\nabla^2\Phi(x^k))^{-1}\nabla\Phi(x^k) + e^k \\
&= x^k - \mathbb{B}^{-1}\nabla\Phi(x^k) + e^k,
\end{aligned}
$$

and

$$x^{k+1} - x^* = x^k - x^* - \mathbb{B}^{-1}\nabla\Phi(x^k) + e^k.$$

Therefore,

$$\mathbb{B}(x^{k+1} - x^*) = \mathbb{B}(x^k - x^*) - \nabla\Phi(x^k) + \mathbb{B}e^k.$$

Now,

$$\|\mathbb{B}e^k\| = \|\nabla^2\Phi(x^k)e^k\| < t\|\nabla\Phi(x^k)\| = t\|\mathbb{B}(x^k - x^*)\|,$$

and

$$\|\mathbb{B}(x^{k+1} - x^*)\| = \|\mathbb{B}e^k\| \le t\|\mathbb{B}(x^k - x^*)\|.$$

$\square$

4 Variants of the general DQN

Let us now discuss the possible alternatives for the choice of $\mathbb{L}_k$. Subsection 4.1 presents three different variants of the general DQN algorithm which mutually differ in the choice of matrix $\mathbb{L}_k$. We refer to the three choices as DQN-0, DQN-1, and DQN-2. All results established in Section 3 hold for these three alternatives. Subsection 4.1 also provides local linear convergence rates for DQN-2 without safeguarding. Subsection 4.2 gives a discussion on the algorithms' tuning parameters, as well as on how the required global knowledge by all nodes can be acquired in a distributed way.

4.1 Algorithms DQN-0, 1, and 2

The analysis presented so far implies only that the diagonal matrix $\mathbb{L}_k$ has to be bounded. Let us now look closer at different possibilities for defining $\mathbb{L}_k$, keeping the restrictions stated in Theorem 3.2 and 3.4.

DQN-0. We first present the method DQN-0 which sets $\mathbb{L}_k = 0$. Clearly, for DQN-0, Theorem 3.2 holds and thus we get linear convergence with the proper choice of ε, and local linear convergence with $\varepsilon = 1$. The quasi Newton matrix approximation in (11) in this case equals $\mathbb{A}_k^{-1}$, i.e., the Hessian is approximated by its block diagonal part only. The method DQN-0 corresponds to Algorithm 1 with only steps 1-4 and 7-9 executed, with $\Lambda_i^k = 0$ in step 7. Clearly, choice $\mathbb{L}_k = 0$ is the cheapest possibility among the choices of $\mathbb{L}_k$ if we consider the computational cost per iteration k. The same holds for communication cost per k, as each node needs to transmit only x_i^k per each iteration, i.e., one p-dimensional vector per node, per iteration is communicated. We note that DQN-0 resembles NN-0, but the difference in general is that DQN-0 uses a different splitting, parameterized with $\theta \geq 0$; actually, NN-0 represents the special case with $\theta = 1$.

DQN-1. Algorithm DQN-1 corresponds to setting $\mathbb{L}_k = \mathbb{L}, k = 0, 1, \ldots,$ where $\mathbb{L}$ is a constant *diagonal* matrix. Assuming that $\mathbb{L}$ is chosen such that $\|\mathbb{L}\| \leq \rho$, with ρ specified in Theorem 3.2, global linear convergence for a proper step size ε and local linear convergence for the full step size $\varepsilon = 1$ again hold. Algorithm DQN-1 is given by Algorithm 1, where each node utilizes a constant, diagonal matrix Λ_i. There are several possible ways of choosing the Λ_i's. In this paper, we focus on the following choice. In the first iteration $k = 0$, each node i sets matrix Λ_i^0 through algorithm DQN-2, stated in the sequel, and then it keeps the same matrix Λ_i^0 throughout the whole algorithm. The computational cost per iteration of DQN-1 is higher than the cost of DQN-0. At each iteration, each node i needs to compute the corresponding inverse of i-th block of $\mathbb{A}_k$ and then to multiply it by the constant diagonal matrix Λ_i. Regarding the communication cost, each node transmits two p-dimensional vectors per iteration – x_i^k and d_i^k (except in the first iteration $k = 0$ when it also transmits vector u_i^0). Although the focus of this paper is on the diagonal $\mathbb{L}_k$'s, we remark that setting $\theta = 1$ and $\mathbb{L}_k = -\mathbb{A}_k^{-1}$ recovers the NN-1 method.

DQN-2. Algorithm DQN-2 corresponds to an iteration-varying, diagonal

17

matrix $\mathbb{L}_k$. Ideally, one would like to choose matrix $\mathbb{L}_k$ such that search direction s^k resembles the Newton step as much as possible, with the restriction that $\mathbb{L}_k$ is diagonal. The Newton direction s_N^k satisfies the equation

$$\nabla^2 \Phi(x^k) s_N^k + \nabla \Phi(x^k) = 0.$$

We seek $\mathbb{L}_k$ such that it makes residual $M(\mathbb{L}_k)$ small, where $M(\mathbb{L}_k)$ is defined as follows:

$$M(\mathbb{L}_k) = \|\nabla^2 \Phi(x^k) s^k + \nabla \Phi(x^k)\|. \tag{28}$$

Notice that

$$
\begin{aligned}
M(\mathbb{L}_k) &= \|\nabla^2 \Phi(x^k) s^k + \nabla \Phi(x^k)\| \\
&= \| - \nabla^2 \Phi(x^k)(I - \mathbb{L}_k \mathbb{G}) \mathbb{A}_k^{-1} \nabla \Phi(x^k) + \nabla \Phi(x^k)\| \\
&= \| - \nabla^2 \Phi(x^k) \mathbb{A}_k^{-1} \nabla \Phi(x^k) + \nabla^2 \Phi(x^k) \mathbb{L}_k \mathbb{G} \mathbb{A}_k^{-1} \nabla \Phi(x^k) + \nabla \Phi(x^k)\| \\
&= \| - (A_k - \mathbb{G}) \mathbb{A}_k^{-1} \nabla \Phi(x^k) + \nabla^2 \Phi(x^k) \mathbb{L}_k \mathbb{G} \mathbb{A}_k^{-1} \nabla \Phi(x^k) + \nabla \Phi(x^k)\| \\
&= \| \mathbb{G} \mathbb{A}_k^{-1} \nabla \Phi(x^k) + \nabla^2 \Phi(x^k) \mathbb{L}_k \mathbb{G} \mathbb{A}_k^{-1} \nabla \Phi(x^k)\|.
\end{aligned}
$$

Therefore,

$$\nabla^2 \Phi(x^k) s^k + \nabla \Phi(x^k) = u^k + \nabla^2 \Phi(x^k) \mathbb{L}_k u^k, \tag{29}$$

where

$$u^k = \mathbb{G} \mathbb{A}_k^{-1} \nabla \Phi(x^k).$$

The minimizer of $M(\mathbb{L}_k)$ is clearly achieved if $\mathbb{L}_k$ satisfies the equation

$$\mathbb{L}_k u^k = -(\nabla^2 \Phi(x^k))^{-1} u^k, \tag{30}$$

but (30) involves the inverse Hessian. Thus we approximate $(\nabla^2 \Phi(x^k))^{-1}$ by the Taylor expansion as follows. Clearly,

$$(\nabla^2 \Phi(x^k))^{-1} = (\alpha \nabla^2 F(x^k) + \mathbb{I} - \mathbb{Z})^{-1} = (\mathbb{I} - \mathbb{V}_k)^{-1}, \quad V_k = \mathbb{Z} - \alpha \nabla^2 F(x^k). \tag{31}$$

Assume that $\alpha < (1 + \lambda_n)/L$, with λ_n being the smallest eigenvalue of W. Then

$$\mathbb{V}_k \succeq (\lambda_n - \alpha L) \mathbb{I} \succ -\mathbb{I}.$$

Similarly,

$$\mathbb{V}_k \preceq (1 - \alpha \mu) \mathbb{I} \prec I.$$

Hence,

$$\rho(\mathbb{V}_k) \leq \|\mathbb{V}_k\|_2 < 1.$$

Therefore, $\mathbb{I} - \mathbb{V}_k$ is nonsingular,

$$(\mathbb{I} - \mathbb{V}_k)^{-1} = \mathbb{I} + \mathbb{V}_k + \sum_{i=2}^{\infty} \mathbb{V}_k^i.$$

and the approximation

$$(\nabla^2 \Phi(x^k))^{-1} = (\mathbb{I} - \mathbb{V}_k)^{-1} \approx \mathbb{I} + \mathbb{V}_k. \tag{32}$$

is well defined. So, we can take $\mathbb{L}_k$ which satisfies the following equation

$$\mathbb{L}_k u^k = -(\mathbb{I} + \mathbb{V}_k)u^k. \tag{33}$$

Obviously, $\mathbb{L}_k$ can be computed in a distributed manner. We refer to the method which corresponds to this choice of $\mathbb{L}_k$ as DQN-2. The algorithm is given by Algorithm 1 where step 6, the choice of $\mathbb{L}_k = diag(\Lambda_1, ..., \Lambda_n)$, involves the steps presented below in Algorithm 2. Denote by u_i^k the i-th $p \times 1$ block of u^k – the block which corresponds to node i.

Algorithm 2

6.1 Each node i calculates

$$u_i^k = \sum_{j \in \bar{O}_i} G_{ij}\, d_j^k.$$

6.2 Each node i transmits u_i^k to all its neighbors $j \in O_i$ and receives u_j^k from all $j \in O_i$.

6.3 Each node i calculates $\widehat{\Lambda}_i^k$ – the solution to the following system of equations (where the only unknown is the $p \times p$ diagonal matrix Λ_i^k):

$$\Lambda_i^k\, u_i^k = -\left[(1 + w_{ii})I - \alpha\, \nabla^2 f_i(x_i^k) \right]\, u_i^k - \sum_{j \in O_i} w_{ij}\, u_j^k.$$

6.4 Each node i projects each diagonal entry of $\widehat{\Lambda}_i^k$ onto the interval $[-\rho, \rho]$.

Note that step 6 with algorithm DQN-2 requires an additional p-dimensional communication per each node, per each k (the communication of the u_i^k's.) Hence, overall, with algorithm DQN-2 each node transmits three p-dimensional vectors per k – x_i^k, d_i^k, and u_i^k.

We next show that algorithm DQN-2 exhibits local linear convergence even when safe-guarding (step 4 in Algorithm 2) is not used.

Theorem 4.1. *Suppose that A1-A3 hold and let x^k be an arbitrary point such that $\nabla\Phi(x^k) \neq 0$. Assume that*

$$\alpha < \min\left\{ \frac{1 + \lambda_n}{L}, \frac{w_{min}}{2L}, \frac{2\mu}{L^2} \right\}, \tag{34}$$

and s^k is generated by (12) and Algorithm 2, Steps 6.1 -6.3. Then there exists $t \in (0,1)$ such that

$$\|\nabla\Phi(x^k) + \nabla^2\Phi(x^k)s^k\| < t\|\nabla\Phi(x^k)\|.$$

Proof. Using (29) and (33) we obtain

$$
\begin{aligned}
\|\nabla^2\Phi(x^k)s^k + \nabla\Phi(x^k)\| &= \|u^k + \nabla^2\Phi(x^k)\mathbb{L}_k u^k\| \\
&= \|u^k - \nabla^2\Phi(x^k)(\mathbb{I} + V_k)u^k\| \\
&= \|(\mathbb{I} - \nabla^2\Phi(x^k)(\mathbb{I} + \mathbb{Z} - \alpha\nabla^2 F(x^k)))u^k\| \\
&= \|\mathbb{P}^k u^k\|, \tag{35}
\end{aligned}
$$

19

where

$$\begin{aligned}
\mathbb{P}^k &= \mathbb{I} - \nabla^2\Phi(x^k)(\mathbb{I} + \mathbb{Z} - \alpha\nabla^2 F(x^k)) \\
&= \mathbb{I} - \left(\mathbb{I} + \alpha\nabla^2 F(x^k) - \mathbb{Z}\right)\left(\mathbb{I} + \mathbb{Z} - \alpha\nabla^2 F(x^k)\right) \\
&= \mathbb{Z}^2 - \alpha(\mathbb{Z}\nabla^2 F(x^k) + \nabla^2 F(x^k)\mathbb{Z}) + \left(\alpha\nabla^2 F(x^k)\right)^2
\end{aligned}$$

Since $\|\nabla^2 f_i(x_i)\| \le L$, there follows $\|\nabla^2 F(x^k)\| \le L$ and the previous equality implies

$$\|\mathbb{P}^k\| \le \|\mathbb{Z}^2 - \alpha(\mathbb{Z}\nabla^2 F(x^k) + \nabla^2 F(x^k)\mathbb{Z})\| + \alpha^2 L^2 := \|U^k\| + \alpha^2 L^2. \quad (36)$$

Now,

$$U_{ij}^k = \sum_{k=1}^{n} w_{ik}w_{kj}I - \alpha(w_{ij}\nabla^2 f_j(x_j^k) + w_{ij}\nabla^2 f_i(x_i^k)).$$

Furthermore, the assumption $\alpha < w_{min}/(2L)$ implies

$$\sum_{k=1}^{n} w_{ik}w_{kj} \ge w_{ii}w_{ij} \ge w_{ij}w_{min} \ge w_{ij}2\alpha L \ge w_{ij}2\alpha\mu.$$

Moreover, $\nabla^2 f_j(x_j^k) \succeq \mu I$ and

$$\|U_{ij}^k\|_2 \le \sum_{k=1}^{n} w_{ik}w_{kj} - 2\alpha\mu w_{ij}.$$

Therefore,

$$\begin{aligned}
\|U^k\| &\le \max_{j=1,\dots,n} \sum_{i=1}^{n}(\sum_{k=1}^{n} w_{ik}w_{kj} - 2\alpha\mu w_{ij}) \\
&= \max_{j=1,\dots,n} \sum_{k=1}^{n} w_{kj} \sum_{i=1}^{n} w_{ik} - 2\alpha\mu \sum_{i=1}^{n} w_{ij} \\
&= 1 - 2\alpha\mu.
\end{aligned}$$

So,

$$\|\mathbb{P}^k\| \le h(\alpha), \quad (37)$$

where $h(\alpha) = 1 - 2\alpha\mu + \alpha^2 L^2$. This function is convex and nonnegative since $\mu \le L$ and therefore

$$\min_{\alpha} h(\alpha) = h\left(\frac{\mu}{L^2}\right) = 1 - \frac{\mu^2}{L^2} > 0.$$

Moreover, $h(0) = h\left(\frac{2\mu}{L^2}\right) = 1$ and we conclude that for all $\alpha \in (0, 2\mu/L^2)$ there holds $h(\alpha) \in (0,1)$. As

$$u^k = \mathbb{G}\mathbb{A}_k^{-1}\nabla\Phi(x^k),$$

we have
$$\|u^k\| \le \|\mathbb{G}\mathbb{A}_k^{-1}\|\|\nabla\Phi(x^k)\|. \tag{38}$$

Now,

$$
\begin{aligned}
\|\mathbb{G}\mathbb{A}_k^{-1}\| &= \max_j(\theta(1-w_{jj})\|(A_j^k)^{-1}\| + \sum_{i\in O_j} w_{ij}\|(A_j^k)^{-1}\|) \\
&\le \max_j \frac{\theta(1-w_{jj})+1-w_{jj}}{\alpha\mu + (1+\theta)(1-w_{jj})} \\
&= \frac{(1+\theta)(1-w_{min})}{\alpha\mu + (1+\theta)(1-w_{min})} < 1.
\end{aligned}
$$

Therefore,
$$\|u^k\| < \|\nabla\Phi(x^k)\|. \tag{39}$$

Putting together (35)-(39), for $\theta \ge 0$ and α satisfying (34) we obtain

$$\|\nabla\Phi(x^k) + \nabla^2\Phi(x^k)s^k\| \le h(\alpha)\|u^k\| < h(\alpha)\|\nabla\Phi(x^k)\|,$$

i.e. the statement holds with

$$t = h(\alpha) = 1 - 2\alpha\mu + \alpha^2 L^2 \tag{40}$$

$\square$

Applying Theorem 3.5 once again, we get the local linear convergence as stated in the following corollary.

Corollary 4.1. *Assume that the conditions of Theorem 4.1 hold. Then there exists η such that for every x^0 satisfying $\|x^0 - x^*\| \le \eta$ the sequence $\{x^k\}$, generated by DQN -2 method with Steps 6.1-6.3 of Algorithm 2 and $\varepsilon = 1$, converges linearly to x^* and*

$$\|\nabla^2\Phi(x^*)(x^{k+1} - x^*)\| \le t\|\nabla^2\Phi(x^*)(x^k - x^*)\|, \quad k = 0, 1, \dots \tag{41}$$

holds with t given by (40).

We remark that, for strongly convex quadratic f_i's, the result analogous to Theorem 3.6 holds in the sense of global linear convergence, i.e., inequality (41) holds for all k and arbitrary initial point x^0.

4.2 Discussion on the tuning parameters

Let us now comment on the choice of tuning parameters – matrix W and scalars $\alpha, \rho, \varepsilon$ and θ. We first consider the general DQN method in Algorithm 1, i.e., our comments apply to all DQN-ℓ variants, $\ell = 0, 1$, and 2.

Matrix W only needs to satisfy that: 1) the underlying support network is connected and 2) all diagonal entries w_{ii} lie between w_{min} and w_{max}, where $0 < w_{min} \le w_{max} < 1$. Regarding the latter condition, it is standard and rather mild; it is only required for (7) to hold, i.e., to ensure that solving (4)

gives an approximate solution to the desired problem (1). Regarding the second condition, it can be easily fulfilled through simple weight assignments, e.g., through the Metropolis weights choice; see, e.g., [32].

The scalar parameters of Algorithm DQN are $\theta, \alpha, \rho, w_{\min}, w_{max}$ and the step size ε. Parameter θ can be arbitrary non-negative constant, and an optimal choice is in general problem-dependent; our numerical experience suggests that taking small values of θ is usually beneficial. Parameter $\alpha > 0$ can be taken arbitrarily, provided that ε is set in accordance with the chosen α, see Theorem 3.3. Recall also (7). Parameter $\rho \geq 0$ needs to be set in accordance with α and θ, as well as with the weight choice parameters – w_{min} and w_{max}, and the f_i's parameters μ and L as specified in (15). The step size ε depends on the above discussed parameters. For global linear convergence, given δ, α, θ, and ρ, parameter ε needs to be set in accordance with μ, L; and w_{min}, w_{max}. For local linear convergence and global linear convergence for strongly convex quadratic f_i's, ε can be taken to unity, provided that the safeguarding parameter ρ is further decreased, see Corollaries 3.2 and 4.1. Finally, with the specific variant DQN-2, one can also set ε to unity and ρ to $+\infty$ (no safeguarding), provided that α is taken sufficiently small relative to a constant depending on w_{min}, μ, and L, see Theorem 4.1 and Corollary 4.1. These conditions guarantee local linear convergence for generic f_i's and global linear convergence for quadratic f_i's.

Notice that, when all the parameters α, θ, δ, and ρ are set as explained above, and ε is set to one (full step-size), global linear convergence in general can not be guaranteed, except for strongly convex quadratic f_i's, but the chosen direction at each iteration k is descent. For practical purposes and faster convergence, the full step-size $\varepsilon = 1$ is tested; our numerical experience on logistic losses suggests that methods DQN-0 and DQN-2 always converged with $\varepsilon = 1$, even without safeguarding on the matrix $\mathbb{L}_k$, while DQN-1 always converged with $\varepsilon = 1$ when the safeguard was included.

Discussion on distributed implementation. The algorithm's tuning parameters need to be set beforehand in a distributed way. Regarding weight matrix W, each node i needs to store beforehand the weights w_{ii} and w_{ij}, $j \in O_i$, for all its neighbors. The weights can be set according to the Metropolis rule, e.g., [32], where each node i needs to know only the degrees of its immediate neighbours. Such weight choice, as noted before, satisfies the imposed assumptions.

In order to set the scalar tuning parameters $\alpha, \theta, \varepsilon$, and ρ, each node i needs to know beforehand global quantities w_{min}, w_{max}, μ and L. Each of these parameters represent either a maximum or a minimum of nodes' local quantities. For example, w_{max} is the maximum of the w_{ii}'s over $i = 1, ..., n$, where node i holds quantity w_{ii}. Hence, each node can obtain w_{max} by running a distributed algorithm for maximum computation beforehand; for example, nodes can utilize the algorithm in [29].

5 Simulations

This section shows numerical performance of the proposed methods on two examples, namely the strongly convex quadratic cost functions and the logistic loss functions.

Simulation setup. Two simulation scenarios with different types of nodes'cost functions f_i's: 1) strongly convex quadratic costs and 2) logistic (convex) loss functions are considered. Very similar scenarios have been considered in [19, 20, 21]. With the quadratic costs scenario, $f_i : \mathbb{R}^p \to \mathbb{R}$ is given by

$$f_i(x) = \frac{1}{2}(x - a_i)^\top B_i(x - a_i),$$

where $B_i \in \mathbb{R}^{p \times p}$ is a positive definite (symmetric matrix), and $a_i \in \mathbb{R}^p$ is a vector. Matrices B_i, $i = 1, ..., n$ are generated mutually independently, and so are the vectors a_i's; also, B_i's are generated independently from the a_i's. Each matrix B_i is generated as follows. First, we generate a matrix $\widehat{B}_i$ whose entries are drawn mutually independently from the standard normal distribution, and then we extract the eigenvector matrix $\widehat{Q} \in \mathbb{R}^{p \times p}$ of matrix $\frac{1}{2}(\widehat{B} + \widehat{B}^\top)$. We finally set $B_i = \widehat{Q}\text{Diag}(\widehat{c}_i)\widehat{Q}^\top$, where $\widehat{c}_i \in \mathbb{R}^p$ has the entries generated mutually independently from the interval $[1, 101]$. Each vector $a_i \in \mathbb{R}^p$ has mutually independently generated entries from the interval $[1, 11]$. Note that a_i–the minimizer of f_i–is clearly known beforehand to node i, but the desired global minimizer of f is not known by any node i.

The logistic loss scenario corresponds to distributed learning of a linear classifier; see, e.g., [1] for details. Each node i possesses $J = 2$ data samples $\{a_{ij}, b_{ij}\}_{j=1}^J$. Here, $a_{ij} \in \mathbb{R}^3$ is a feature vector, and $b_{ij} \in \{-1, +1\}$ is its class label. We want to learn a vector $x = (x_1^\top, x_0)^\top$, $x_1 \in \mathbb{R}^{p-1}$, and $x_0 \in \mathbb{R}$, $p \geq 2$, such that the total logistic loss with l_2 regularization is minimized:

$$\sum_{i=1}^N \sum_{j=1}^J \mathcal{J}_{\text{logis}}\left(b_{ij}(x_1^\top a + x_0)\right) + \tau\|x\|^2,$$

Here, $\mathcal{J}_{\text{logis}}(\cdot)$ is the logistic loss

$$\mathcal{J}_{\text{logis}}(z) = \log(1 + e^{-z}),$$

and τ is a positive regularization parameter. Note that, in this example, we have

$$f_i(x) = \sum_{j=1}^J \mathcal{J}_{\text{logis}}\left(b_{ij}(x_1^\top a + x_0)\right) + \frac{\tau}{n}\|x\|^2,$$

$f(x) = \sum_{i=1}^N f_i(x)$. The a_{ij}'s are generated independently over i and j, where each entry of a_{ij} is drawn independently from the standard normal distribution. The "true" vector $x^\star = ((x_1^\star)^\top, x_0^\star)^\top$ is obtained by drawing its entries independently from standard normal distribution. Then, the class labels are

$b_{ij} = \text{sign}\left((x_1^\star)^\top a_{ij} + x_0^\star + \varepsilon_{ij}\right)$, where ε_{ij}'s are drawn independently from normal distribution with zero mean and standard deviation 0.1.

The network instances are generated from the random geometric graph model: nodes are placed uniformly at random over a unit square, and the node pairs within distance $r = \sqrt{\frac{\ln(n)}{n}}$ are connected with edges. All instances of networks used in the experiments are connected. The weight matrix W is set as follows. For a pair of nodes i and j connected with an edge, $w_{ij} = \frac{1}{2\max\{d_i, d_j\}+1}$, where d_i is the degree of the node i; for a pair of nodes not connected by an edge, we have $w_{ij} = 0$; and $w_{ii} = 1 - \sum_{j \neq i} w_{ij}$, for all i. For the case of regular graphs, considered in [19, 20, 21], this weight choice coincides with that in [19, 20, 21].

The proposed methods DQN are compared with the methods NN-0, NN-1, and NN-2 proposed in [19]. The methods NN-ℓ, with $\ell \geq 3$ are not numerically tested in [19] and require a large communication cost per iteration. Recall that the method proposed in this paper are denoted DQN-ℓ with $\mathbb{L}_k = 0$ as DQN-0; it has the same communication cost per iteration k as NN-0, where each node transmits one (p-dimensional) vector per iteration. Similarly, DQN-1 corresponds to NN-1, where two per-node vector communications are utilized, while DQN-2 corresponds to NN-2 (3 vector communications per node).

With both the proposed methods and the methods in [19], the step size $\varepsilon = 1$ is used. Step size $\varepsilon = 1$ has also been used in [19, 20, 21]. Note that both classes of methods – NN and DQN – guarantee global convergence with $\varepsilon = 1$ for quadratic costs, while neither of the two groups of methods have guaranteed global convergence with logistic losses. For the proposed methods, safeguarding is not used with quadratic costs. With logistic costs, the safeguarding is not used with DQN-0 and 2 but it is used with DQN-1, which diverges without the safeguard on the logistic costs. The safeguard parameter ρ defined in Theorem 3.2, with $\delta = 0$ is employed. Further, with all DQNs, $\theta = 0$ is used. With all the algorithms, each node's solution estimate is initialized by a zero vector.

The following error metric

$$\frac{1}{n} \sum_{i=1}^{n} \frac{\left\| x_i^k - x^\star \right\|}{\|x^\star\|}, \ x^\star \neq 0,$$

is used and refered to as the relative error at iteration k.

Figure 1 (left) plots the relative error versus the number of iterations k for a network with $n = 30$ nodes, and the quadratic costs with the variable dimension $p = 4$. First, we can see that the proposed DQN-ℓ methods perform better than their corresponding counterparts NN-ℓ, $\ell = 0, 1, 2$. Also, note that the performance of DQN-1 and DQN-2 in terms of iterations match in this example. Figure 1 (right) plots the relative error versus total number of communications. We can see that, for this example, DQN-0 is the most efficient among all methods in terms of the communication cost. Further, interestingly, the performance of NN-0, NN-1, and NN-2 is practically the same in terms of communication cost on this example. The clustering of the performance of NN-0, NN-1, and

NN-2 (although not so pronounced as in our examples) emerges also in Simulations in [19, 20, 21]. In summary, method DQN-0 shows the best performance in terms of communication cost on this example, while DQN-1 and 2 are the best in terms of the number of iterations k.

Figure 2 (left and right) repeats the plots for the network with $n = 400$ nodes, quadratic costs, and the variable dimension $p = 3$. One can see that again the proposed methods outperform their respective NN-ℓ counterparts. In terms of communication cost, DQN-0 and DQN-1 perform practically the same and are the most efficient among all methods.
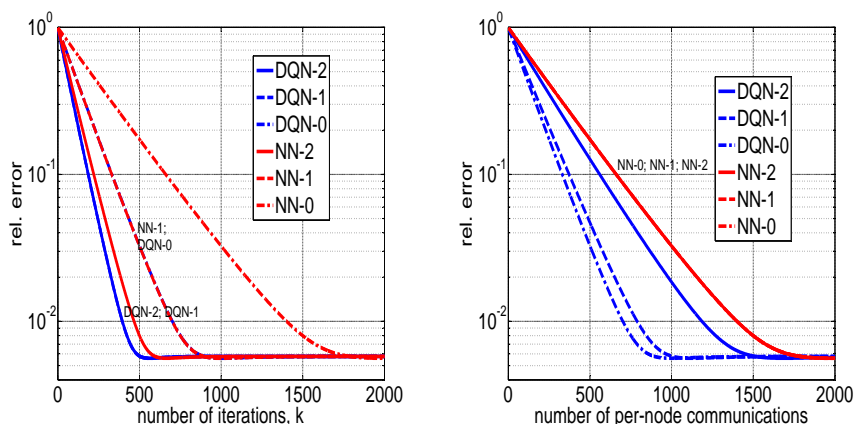
Figure 1: Relative error versus number of iterations k (left) and versus number of communications (right) for quadratic costs and $n = 30$-node network.

Figure 3 plots the relative error versus number of iterations (left) and number of per-node communications (right) for the logistic losses with variable dimension $p = 4$ and the network with $n = 30$ nodes. One can see that again the proposed methods perform better than the NN-ℓ counterparts. In terms of the communication cost, DQN-0 is the most efficient among all methods, while DQN-2 is fastest in terms of the number of iterations. Finally, Figure 4 repeats the plots for variable dimension $p = 4$ and the network with $n = 200$ nodes, and it shows similar conclusions: among all DQN and NN methods, DQN-0 is the most efficient in terms of communications, while DQN-2 is fastest in terms of the number of iterations.

6 Conclusions

The problem under consideration is defined by an aggregate cost function on a simple connected network. It is assumed that the cost functions are convex an differentiable while the network is characterized by a double stochastic matrix W that fulfils the standard assumptions. The proposed methods are designed
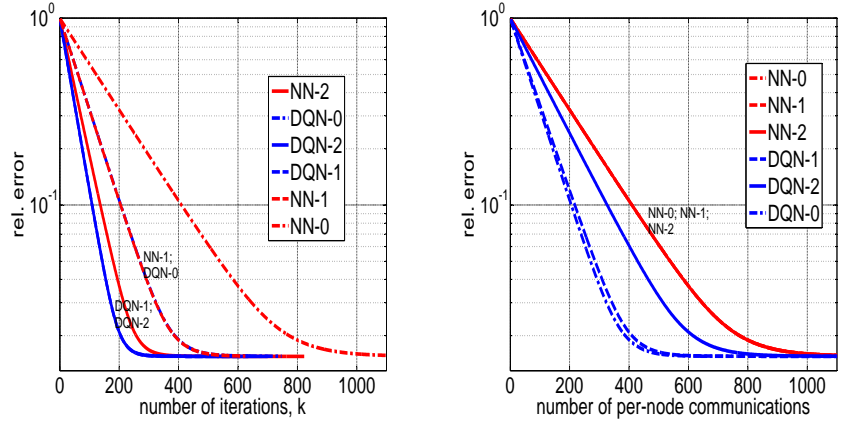
Figure 2: Relative error versus number of iterations k (left) and versus number of communications (right) for quadratic costs and $n = 400$-node network.
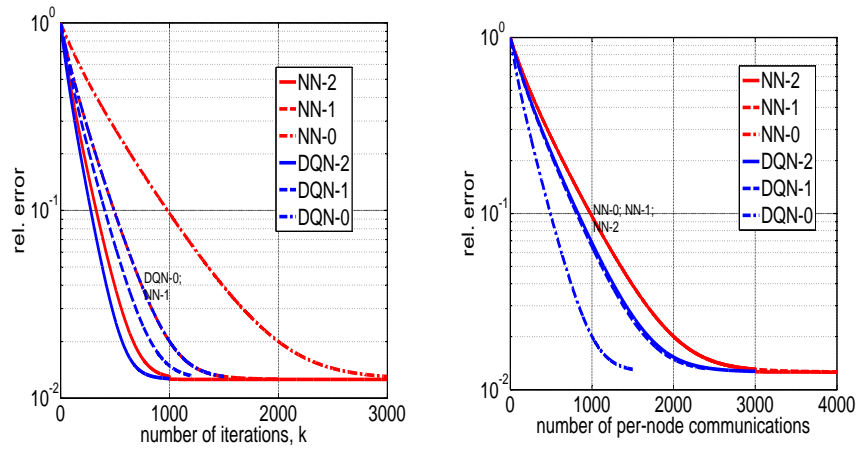

Figure 3: Relative error versus number of iterations k (left) and versus number of communications (right) for logistic costs and $n = 30$-node network.
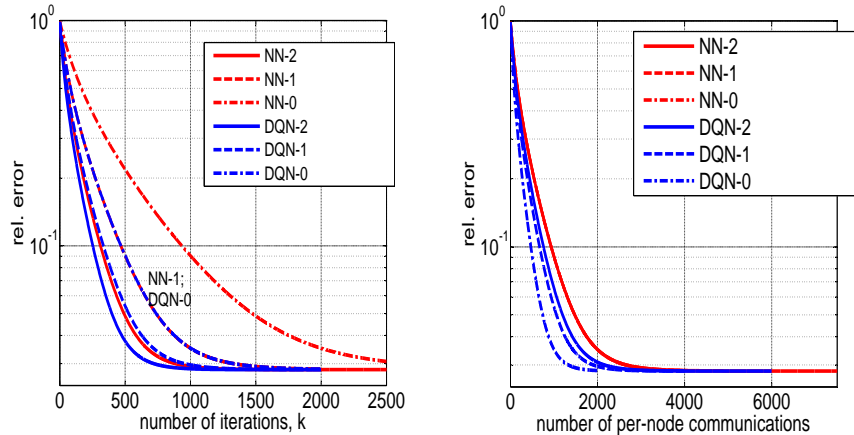
Figure 4: Relative error versus number of iterations k (left) and versus number of communications (right) for logistic costs and $n = 200$-node network.

for the penalty reformulation of the original problem and rely heavily on the sparsity structure of the Hessian. The general method is tailored as a Newton-like method, taking the block diagonal part of the Hessian as an approximate Hessian and then correcting this approximation by a diagonal matrix. Thus a particular method is specified by the diagonal matrix in the Hessian approximation. The key point in the proposed class of methods is to exploit the structure of the Hessian and to replace the dense part of the inverse Hessian by a inexpensive linear approximation, determined by the diagonal matrix. Depending on the choice of this diagonal matrix one can define different methods and three of these choices are analysed in the work presented here. An important characteristic of the whole class of DQN methods is global linear convergence with a proper choice of the step size. Furthermore, we have shown local linear convergence for the full step size using the convergence theory of Inexact Newton methods as well as global convergence with the full step size for the special case of strictly convex quadratic loss functions.

Three particular methods are analysed in detail, DQN-0, DQN-1 and DQN-2. They are defined by the three different choices of the diagonal matrix - the zero matrix, a constant matrix and the iteration-varying matrix that defines a search direction which mimics the Newton direction as much as possible under the imposed restrictions of distributional and inexpensive computation. For the last choice of time varying matrix we have shown local linear convergence for the full step size without the safeguard.

The cost in terms of computational effort and communication of these three methods correspond to the costs of the state-of-the-art Network Newton methods, NN-0, NN-1 and NN-2, which are used as the benchmark class in this paper. The simulation results on two relevant problems, the quadratic loss and the logistic loss, demonstrate the efficiency of the proposed methods and compare

27

favourably with the benchmark methods.

References

[1] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning, Volume 3, Issue 1, (2011) pp. 1-122.

[2] Byrd, R. H., Chin, G. M., Neveitt, W., Nocedal, J., On the Use of Stochastic Hessian Information in Optimization Methods for Machine Learning, *SIAM Journal on Optimization*, 21 (3), (2011) pp. 977-995.

[3] Byrd, R. H.. Chin, G. M., Nocedal, J., Wu, Y., Sample size selection in optimization methods for machine learning, *Mathematical Programming*, no. 134 vol. 1 (2012) pp. 127-155.

[4] Byrd, R. H., Hansen, S. L. , Nocedal, J., Singer, Y., A Stochastic Quasi-Newton Method for Large-Scale Optimization, *available at arXiv:1401.7020 [math.OC]*.

[5] Cattivelli, F., Sayed, A. H., Diffusion LMS strategies for distributed estimation, *IEEE Transactions on Signal Processing*, vol. 58, no. 3, (2010) pp. 1035–1048.

[6] Chen I.-A., Ozdaglar, A., A fast distributed proximal gradient method, in *Allerton Conference on Communication, Control and Computing*, Monticello, IL, October 2012.

[7] Daneshmand, A., Facchinei, F., Kungurtsev, V., Scutari, G., Hybrid random/deterministic parallel algorithms for nonconvex big data optimization, *submitted to IEEE Transactions on Signal Processing*, 2014.

[8] Dembo, R.S. Eisenstat, S.C. Steinhaug, T., Inexact Newton method, *SIAM Journall on Numerical Analysis* 19,2, (1982), 400-409.

[9] Feingold, D.G., Varga, R. S., Block Diagonally Dominant Matrices and Generalizations of the Gershgorin Circle Theorem, *Pacific Journal of Mathematics*, 12 (4), (1962) 1241-1250.

[10] Friedlander, M. P. , Schmidt, M., Hybrid deterministic-stochastic methods for data fitting, *SIAM Journal on Scientific Computing* 34 (2012), 1380-1405.

[11] Jakovetić, D., Xavier, J., Moura, J. M. F., Fast distributed gradient methods, *IEEE Transactions on Automatic Control*, vol. 59, no. 5, (2014) pp. 1131–1146.

[12] Jakovetić, D., Moura, J. M. F., Xavier, J., Distributed Nesterov-like gradient algorithms, in *CDC'12, 51st IEEE Conference on Decision and Control*, Maui, Hawaii, December 2012, pp. 5459–5464.

[13] Kar, S., Moura, J. M. F., Ramanan, K., Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication, *IEEE Transactions on Information Theory*, vol. 58, no. 6, (2012) pp. 3575–3605.

[14] Krejić, N., Krklec, N., Variable sample size methods for unconstrained optimization, *Journal of Computational and Applied Mathematics* 245 (2013), 213-231.

[15] Krejić, N., Krklec Jerinkić, N., Nonmonotone line search methods with variable sample size, *Numerical Algorithms* 68 (2015), pp. 711-739.

[16] Krejić N., Lužanin Z., Newton-like method with modification of the right-hand side vector, *Mathematics of Computation*, 71, 237 (2002), 237-250.

[17] Lobel, I., Ozdaglar, A., Feijer, D., Distributed Multi-agent Optimization with State-Dependent Communication, *Mathematical Programming*, vol. 129, no. 2, (2014) pp. 255-284.

[18] Lopes, C., Sayed, A. H., Adaptive estimation algorithms over distributed networks, in *21st IEICE Signal Processing Symposium*, Kyoto, Japan, Nov. 2006.

[19] Mokhtari, A., Ling, Q., Ribeiro, A., An approximate Newton method for distributed optimization, 2014, available at http://www.seas.upenn.edu/ aryanm/wiki/NNICASSP.pdf

[20] Mokhtari, A., Ling, Q., Ribeiro, A., Network Newton–Part I: Algorithm and Convergence, 2015, available at: http://arxiv.org/abs/1504.06017

[21] Mokhtari, A., Ling, Q., Ribeiro, A., Network Newton–Part II: Convergence Rate and Implementation, 2015, available at: http://arxiv.org/abs/1504.06020

[22] Mokhtari, A., Ribeiro, A., Regularized Stochastic BFGS method, *IEEE Transactions on Signal Processing*, Vol. 62, no. 23, (2014) pp. 6089-6104.

[23] Mokhtari, A., Ribeiro, A., Global convergence of Online Limited Memory BFGS Method, J. Machine Learning Research, vol. (revised), April 2015.

[24] Mota, J., Xavier, J., Aguiar, P., Püschel, M., Distributed optimization with local domains: Applications in MPC and network flows, *to appear in IEEE Transactions on Automatic Control*, 2015.

[25] Nedić, A., Ozdaglar, A., Distributed subgradient methods for multi-agent optimization, *IEEE Transactions on Automatic Control*, vol. 54, no. 1, (2009) pp. 48–61.

[26] Ram, S., Nedić, A., Veeravalli, V., Distributed stochastic subgradient projection algorithms for convex optimization, *Journal on Optimization Theory and Applications*, vol. 147, no. 3, (2011) pp. 516–545.

[27] Ram, S. S. , Nedić, A., Veeravalli, V., Asynchronous gossip algorithms for stochastic optimization, in *CDC '09, 48th IEEE International Conference on Decision and Control*, Shanghai, China, December 2009, pp. 3581 – 3586.

[28] Schizas, I. D. , Ribeiro, A., Giannakis, G. B., Consensus in ad hoc WSNs with noisy links – Part I: Distributed estimation of deterministic signals, *IEEE Transactions on Signal Processing*, vol. 56, no. 1, (2009) pp. 350–364.

[29] Shi, G., Johansson, K. H., Finite-time and asymptotic convergence of distributed averaging and maximizing algorithms, 2012, available at: http://arxiv.org/pdf/1205.1733.pdf

[30] Shi, W., Ling, Q., Wu, G., Yin, W., EXTRA: an Exact First-Order Algorithm for Decentralized Consensus Optimization, SIAM Journal on Optimization No. 25 vol. 2, (2015) pp. 944-966.

[31] Wei, E., Ozdaglar, A., Jadbabaie, A., A distributed Newton method for network utility maximizationI: algorithm, *IEEE Transactions on Automatic Control,* vol. 58, no. 9, (2013) pp. 2162- 2175.

[32] Xiao, L., Boyd, S., Lall, S., A scheme for robust distributed sensor fusion based on average consensus, in *IPSN '05, Information Processing in Sensor Networks*, Los Angeles, California, 2005, pp. 63–70.

[33] Yuan, L., Ling, Q., Yin, W., On the convergence of decentralized gradient descent, 2013, available at: http://arxiv.org/abs/1310.7063.

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