NEWTON-LIKE METHOD WITH DIAGONAL CORRECTION FOR DISTRIBUTED OPTIMIZATION[*]

DRAGANA BAJOVIĆ[†], DUŠAN JAKOVETIĆ[†], NATAŠA KREJIĆ[‡], AND
NATAŠA KRKLEC JERINKIĆ[‡]

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 parts, (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 demonstrate the effectiveness of the proposed DQN methods.

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

AMS subject classifications. 90C25, 90C53, 90K05

DOI. 10.1137/15M1038049

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

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

Problems of this form arise in many emerging applications like big data analytics, e.g., [7], distributed inference in sensor networks [33, 16, 21, 5], and distributed control [29].

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 [30, 31, 32, 14, 15, 6, 40, 36, 20]. 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

[†]Biosense Institute, University of Novi Sad, Zorana Djindjića 3, 21000 Novi Sad, Serbia (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 (natasak@uns.ac.rs, natasa.krklec@dmi.uns.ac.rs).

the trade-off between the convergence speed toward a solution neighborhood and the distance of the limit point from the actual solution, and larger α means faster convergence but larger distance from the solution in the limit; see, e.g., [15], [22]. 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 [15], [22] 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 [12, 17, 18, 25, 26].

There have been several papers on distributed Newton-type methods. Distributed second order methods for network utility maximization and network flow optimization are developed in [37] and [41] but on problem formulations different from (1). The network Newton (NN) method [22] aims at solving (1) and presents a family of distributed (approximate) Newton methods. The class of NN methods is extensively analyzed in [23, 24]. The proposed methods are based on the penalty interpretation [15, 22] of the distributed gradient method in [30], and they approximate the Newton step through an ℓth 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 trade-offs 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 [30]. Reference [27] proposes a distributed second order method which approximates the (possibly expensive) primal updates with the distributed alternating direction of multipliers method in [35]. In [38], the authors propose a distributed Newton Raphson method based on the consensus algorithm and separation of time-scales. Reference [28] proposes distributed second order methods based on the proximal method of multipliers (PMM). Specifically, the methods approximate the primal variable update step through a second order approximation of the NN-type [22]. While the NN methods in [22] converge to a solution neighborhood, the methods in [28, 27, 38] converge to *exact* solutions.

In this paper, we extend [22, 23, 24] and propose a different family of distributed Newton-like methods for solving (1). We refer to the proposed family as distributed quasi-Newton (DQN) methods. The methods are designed to exploit the specific structure of the penalty reformulation [15, 22], as is done in [22], but with a different Hessian inverse approximation, for which the idea originates in [19]. 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. The methods exhibit linear convergence to a solution neighborhood 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 favorably with NN proposed in [22].

With the DQN family of methods, the approximation of the Newton step is parameterized by diagonal matrix $\mathbb{L}_k$ at 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. Different variants of DQN, based on different matrices $\mathbb{L}_k$, trade off the number of iterations and computational cost. In particular, setting $\mathbb{L}_k = 0$ yields the DQN-0 method; a constant, diagonal matrix $\mathbb{L}_k = \mathbb{L}$ corresponds to DQN-1. Finally, $\mathbb{L}_k$ with DQN-2 is obtained through approximately fitting the Newton equation (34) using a first order Taylor approximation. The DQN-1 method utilizes the latter, DQN-2's weight matrix at the first iteration, and then it "freezes" it to this constant value throughout the iterations; that is, it sets $\mathbb{L} = \mathbb{L}_0$, where $\mathbb{L}_0$ corresponds to the weight matrix of DQN-2 in the initial iteration.

Let us further specify the main differences between the proposed DQN family and NN methods in [22] as the NN methods are used as the benchmark in the work presented here. First, the DQN methods introduce a different, more general splitting of Hessians with respect to NN, parameterized with a scalar $\theta \geq 0$; in contrast, the splitting used in NN corresponds to setting $\theta = 1$. Second, with the proposed variants of DQN-0, 1, and 2, we utilize *diagonal matrices* $\mathbb{L}_k$, while the NN-ℓ methods use in general block-diagonal or neighbor-sparse matrices. Third, DQN and NN utilize different inverse Hessian approximations; the NN's inverse Hessian approximation matrix is symmetric, while with DQN it is not symmetric in general. Fourth, while NN approximates the inverse Hessian directly (independently of the Newton equation), DQN actually aims at approximating the Newton equation. Hence, unlike NN, the resulting DQN's inverse Hessian approximation (with DQN-2 in particular) explicitly depends on the gradient at the current iterate, as is the case with many quasi-Newton methods; see, e.g., [9]. Finally, the analysis here is very different from [22], and the major reason comes from the fact that the Hessian approximation with DQN is asymmetric 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 [22] 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. It may be very challenging to rigorously compare the linear convergence rate factors of DQN and NN methods and their respective inverse Hessian approximations. However, we provide in section 5 both certain analytical insights and extensive numerical experiments to compare the two classes of methods.

As noted, DQN methods do not converge to the exact solution of (1), but they converge to a solution neighborhood, as is the case with other methods (e.g., distributed gradient descent [30] and NN methods [22]) which are based on the penalty interpretation of (1). Hence, for very high accuracies, they may not be competitive with distributed second order methods which converge to the exact solution [27, 28, 38]. However, following the framework of embedding distributed second order methods into PMM, developed in [28], we apply here the DQN Newton direction approximations to the PMM methods; we refer to the resulting methods as PMM-DQN-ℓ, $\ell = 0, 1, 2$. Simulation examples on strongly convex quadratic costs demonstrate that the PMM-DQN methods compare favorably with the methods in [27, 28, 38]. Therefore, with respect to the existing literature and in particular with respect to [22, 28], this paper broadens the possibilities for distributed approximations of relevant Newton directions and hence offers alternative distributed second order methods, which

exhibit competitive performance on the considered simulation examples. Analytical studies of PMM-DQN are left for future work.

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 methods 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 section 6 discusses extensions of embedding DQN in the PMM framework. Finally, conclusions are drawn in section 7, while the appendix provides some auxiliary derivations.

2. Preliminaries. Let us first give some preliminaries about problem (1), its penalty interpretation in [15, 22], as well as the decentralized gradient descent algorithm in [30] 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 $0 < \mu \le L < \infty$ 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 $M \preceq N$ means that the matrix $N - M$ is positive semidefinite.

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

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

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

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

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, (doubly) 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

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

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

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

Denote by $\lambda_1 \ge \cdots \ge \lambda_n$ 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 [15], [22], 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

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

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

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

which, denoting the ith $p \times 1$ block of x^k by x_i^k, and after rearranging terms, yields the decentralized gradient descent method [30]

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

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 [22] 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,

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

The convergence of (6) toward x^* is linear, i.e., the following estimate holds [40, 15]:

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

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

The matrix and vector norms that will be frequently used in what follows are defined here. Let $\|a\|_2$ denote the Euclidean norm of vector a of arbitrary dimension. Next, $\|A\|_2$ denotes the spectral norm of matrix A of arbitrary dimension. Further, for a matrix $\mathbb{M} \in \mathbb{R}^{np \times np}$ with blocks $M_{ij} \in \mathbb{R}^{p \times p}$, we will also use the following block norm:

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

where, as noted, $\|M_{ij}\|_2$ is the spectral norm of M_{ij}. For a vector $x \in \mathbb{R}^{np}$ with blocks $x_i \in \mathbb{R}^p$, the following block norm is also used:

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

3. Distributed quasi-Newton method. In this section we introduce a class of quasi-Newton methods for solving (4). The general DQN method is proposed in subsection 3.1. The method is characterized by a generic diagonal matrix $\mathbb{L}_k$. The 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 [22], 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 nondiagonal part of $\nabla^2 \Phi(x)$ to correct the right-hand-side vector in the (quasi-) Newton equation. Let us define the splitting

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

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

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

Here, $\mathrm{diag}(W)$ denotes the diagonal matrix with the same diagonal as the matrix W. Hence, matrix $\mathbb{Z}_d$ is an $np \times np$ diagonal matrix whose ith $p \times p$ block is the scalar matrix $w_{ii}I$, $\mathbb{Z}_u$ is an $np \times np$ 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

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

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 DQN algorithm is presented below. Denote by k the iteration counter, $k = 0, 1, \dots$, and let $x^k = (x_1^k, \dots, x_n^k) \in \mathbb{R}^{np}$ be the estimate of x^* at iteration k. Consider the following splitting of the Hessian:

$$(9) \qquad \nabla^2 \Phi(x^k) = \mathbb{A}_k - \mathbb{G}$$

with

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

and

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

for some $\theta \geq 0$. Hence, $\mathbb{G}$ is an $np \times np$ block (symmetric) matrix whose ith $p \times p$ 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 $g_{ij} := w_{ij}$. One can easily see that the splitting above recovers the splitting for NN methods [22] taking $\theta = 1$. We keep θ unspecified for now, and later on we will demonstrate numerically that taking $\theta = 0$ can be beneficial. 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 diagonal matrix composed of diagonal $p \times p$ matrices Λ_i^k, $i = 1, \ldots, n$. In this paper, we adopt the following approximation of the Newton direction $s_N^k = -(\mathbb{A}_k - \mathbb{G})^{-1} \nabla \Phi(x^k)$:

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

The motivation for this approximation comes from [19] and the following reasoning. Keep the Hessian approximation on the left-hand side of the Newton equation $(\mathbb{A}_k - \mathbb{G}) s_N^k = -\nabla \Phi(x^k)$ diagonal, and correct the right-hand side through the off-diagonal Hessian part. In more detail, the Newton equation can be equivalently written as

(12) $$\mathbb{A}_k s_N^k = -\nabla \Phi(x^k) + \mathbb{G} s_N^k,$$

where the off-diagonal part $\widetilde{s}^k := \mathbb{G} s_N^k$ is moved to the right-hand side. If we pretended for a moment to know the value of $\widetilde{s}^k$, then the Newton direction is obtained as

(13) $$s_N^k = -(\mathbb{A}_k)^{-1} \left(\nabla \Phi(x^k) - \widetilde{s}^k \right).$$

This form is suitable for distributed implementation due to the need to invert only the block diagonal matrix $\mathbb{A}_k$. However, $\widetilde{s}^k$ is clearly not known, and hence we approximate it. To this end, note that, assuming that $\mathbb{G} \nabla \Phi(x^k)$ is a vector with all the entries being nonzero, without loss of generality, $\widetilde{s}^k$ can be written as follows:

(14) $$\widetilde{s}^k = \mathbb{L}_k \mathbb{G} \nabla \Phi(x^k),$$

where $\mathbb{L}_k$ is a *diagonal matrix*. Therefore, estimating $\widetilde{s}^k$ translates into estimating the diagonal matrix $\mathbb{L}_k$, assuming that $\mathbb{G} \nabla \Phi(x^k)$ is known. We follow [19] and consider simple approximations of the "true" $\mathbb{L}_k$. For example, we will consider $\mathbb{L}_k = 0$, which discards the off-diagonal Hessian part. Also, as we will see ahead, we adopt a Taylor approximation method for estimating $\mathbb{L}_k$. Now, substituting (14) into (13), we obtain the following Newton direction approximation:

(15) $$s_0^k = -(\mathbb{A}_k)^{-1} (\mathbb{I} - \mathbb{L}_k \mathbb{G}) \nabla \Phi(x^k).$$

Finally, we adopt (11) as the definite form of the Newton direction approximation, i.e., we permute the matrices $(\mathbb{I} - \mathbb{L}_k \mathbb{G})$ and $(\mathbb{A}_k)^{-1}$. The reason is that both s^k and s_0^k have the same inner product with $\nabla \Phi(x^k)$, and hence they have the same descent properties (for example, Theorem 3.2 ahead holds unchanged for s^k and s_0^k), while a careful inspection shows that s^k allows for a cheaper (in terms of communications per iteration) distributed implementation.

The reasoning above justifies restriction to *diagonal* $\mathbb{L}_k$'s, i.e., in view of (14), adopting diagonal $\mathbb{L}_k$'s does not *in principle, i.e., in structure* sacrifice the quality of the Newton direction approximation, while it is computationally cheap.

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

$$(16) \qquad x^{k+1} = x^k + \varepsilon s^k$$

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) and the search direction (11). For this moment we are assuming only that θ is nonnegative and fixed initially, while further details are presented later on.

In summary, the proposed distributed algorithm for solving (4) is given below.

Algorithm 1: DQN in vector format. Given $x^0 \in \mathbb{R}^{np}, \varepsilon, \rho > 0$. Set $k = 0$.
Step 1. Chose a diagonal matrix $\mathbb{L}_k \in \mathbb{R}^{np \times np}$ such that

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

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 2.

Algorithm 2: DQN—distributed implementation. At each node i, require $\rho, \varepsilon > 0$.

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 $\|\Lambda_i^k\|_2 \le \rho$.
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 interneighbor communication of a p-dimensional vector. Likewise, choices of tuning parameters $\varepsilon, \rho, \theta$ are discussed throughout the rest of this section and section 4.

Remark 3.1. It is useful to compare (11) with the direction adopted 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 cannot be recovered in a similar fashion. Hence, DQN in a sense generalizes NN-0 and NN-1. An approximation $\mathbb{L}_k = \mathbb{L}$, which will be considered later on, does not recover any NN methods.

Observe that the approximation matrix $(\mathbb{I} - \mathbb{L}_k \mathbb{G})\mathbb{A}_k^{-1}$ is not symmetric in general. This fact induces the need for a safeguarding step in Algorithm 1; more precisely the elements of $\mathbb{L}_k$ are uniformly bounded as stated in step 1 (Algorithm 1), $\|\mathbb{L}\| \le \rho$, and the resulting method requires an analysis different from [22].

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 3.2 (see [11]). *For any $\mathbb{C} \in \mathbb{R}^{np \times np}$ partitioned into blocks C_{ij} of size $p \times p$, each eigenvalue μ of $\mathbb{C}$ satisfies*

$$(17) \qquad \frac{1}{\|(C_{ii} - \mu I)^{-1}\|_2} \le \sum_{i \ne j} \|C_{ij}\|_2$$

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

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

COROLLARY 3.3. *Let $\mathbb{C} \in \mathbb{R}^{np \times np}$ be a symmetric matrix partitioned into blocks C_{ij} of size $p \times p$. Then each eigenvalue μ of $\mathbb{C}$ satisfies*

$$\mu \ge \min_{i=1,\dots,n} \left\{ \lambda_{\min}(C_{ii}) - \sum_{j \ne i} \|C_{ij}\|_2 \right\},$$

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 3.2, there exists $i \in \{1, \dots, n\}$, such that (17) holds. Next, we have

$$\|(C_{ii} - \mu I)^{-1}\|_2 = \frac{1}{\min_{j=1,\dots,p} |\lambda_j(C_{ii}) - \mu|},$$

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

$$\min_{j=1,\dots,p} |\lambda_j(C_{ii}) - \mu| \le \sum_{j \ne i} \|C_{ij}\|_2.$$

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

$$[\lambda_j(C_{ii}) - \sum_{i \ne l} \|C_{il}\|_2, \lambda_j(C_{ii}) + \sum_{i \ne l} \|C_{il}\|_2].$$

Hence, for each μ for which (17) holds for some fixed i, we have

$$\mu \geq \lambda_{\min}(C_{ii}) - \sum_{l \neq i} \|C_{il}\|_2$$

and the statement follows.   □

We are now ready to prove that the search direction (11) is descent.

THEOREM 3.4. *Suppose that* A1–A3 *hold. Let*

$$(18) \qquad 0 \leq \rho \leq \frac{\alpha\mu + (1+\theta)(1 - w_{max})}{(1 - w_{min})(1 + \theta)} \left(\frac{1}{\alpha L + (1+\theta)(1 - w_{min})} - \delta \right)$$

for some $\delta \in (0, 1/(\alpha L + (1+\theta)(1 - w_{min})))$. *Then* s^k *defined by* (11) *is a descent direction and satisfies*

$$\nabla^T \Phi(x^k) s^k \leq -\delta \|\nabla \Phi(x^k)\|_2^2.$$

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 should be 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 3.3 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

$$(19) \qquad (\alpha\mu + (1+\theta)(1 - w_{ii}))I \preceq A_i^k \preceq (\alpha L + (1+\theta)(1 - w_{ii}))I$$

so

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

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 that

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

and

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

Now,

$$\lambda_{min}(C^k) \geq \min_{i=1,\ldots,n} \left(\frac{1}{\alpha L + (1+\theta)(1-w_{min})} - \frac{\theta(1-w_{ii})\rho}{\alpha\mu + (1+\theta)(1-w_{max})} \right.$$

$$\left. - \sum_{j\in O_i} w_{ij} \frac{\rho}{\alpha\mu + (1+\theta)(1-w_{max})} \right)$$

$$= \min_{i=1,\ldots,n} \left(\frac{1}{\alpha L + (1+\theta)(1-w_{min})} - \frac{\rho(1-w_{ii})(1+\theta)}{\alpha\mu + (1+\theta)(1-w_{max})} \right)$$

$$\geq \frac{1}{\alpha L + (1+\theta)(1-w_{min})} - \rho\frac{(1-w_{min})(1+\theta)}{\alpha\mu + (1+\theta)(1-w_{max})}$$

$$(20) \qquad \geq \delta.$$

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

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

$$= -\nabla^T\Phi(x^k)C^k\nabla\Phi(x^k)$$

$$(21) \qquad \leq -\delta\|\nabla\Phi(x^k)\|_2^2. \qquad \square$$

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

LEMMA 3.5. *Suppose that* A1–A3 *hold. Then*

$$\|s^k\|_2 \leq \beta\|\nabla\Phi(x^k)\|_2,$$

where

$$(22) \qquad \beta = \frac{1 + \rho(1+\theta)(1-w_{min})}{\alpha\mu + (1+\theta)(1-w_{max})}.$$

Proof. For matrix $\mathbb{A}_k$, there holds that

$$(23) \qquad \|\mathbb{A}_k^{-1}\|_2 \leq \frac{1}{\alpha\mu + (1+\theta)(1-w_{max})}.$$

This can be shown similarly as the upper bound on $\|(A_i^k)^{-1}\|_2$ below (19). Furthermore,

$$(24) \qquad \|\mathbb{G}\|_2 \leq (1+\theta)(1-w_{min}).$$

This is true as

$$\|\mathbb{G}\|_2 = \|\mathbb{Z}_u + \theta(\mathbb{I} - \mathbb{Z}_d)\|_2 \le \|\mathbb{Z}_u\|_2 + \theta\|\mathbb{I} - \mathbb{Z}_d\|_2$$
$$\le \|\mathbb{I} - \mathbb{Z}_d\|_2 + \theta\|\mathbb{I} - \mathbb{Z}_d\|_2 \le (1 + \theta)(1 - w_{min}),$$

where we used $\mathbb{Z}_u = \mathbb{Z} - \mathbb{Z}_d \preceq \mathbb{I} - \mathbb{Z}_d \preceq (1 - w_{min})\mathbb{I}$. Therefore, we have

$$\|s^k\|_2 \le \|(\mathbb{I} - \mathbb{L}_k\mathbb{G})\mathbb{A}_k^{-1}\|_2\|\nabla\Phi(x^k)\|_2$$
$$\le (1 + \|\mathbb{L}_k\|_2\|\mathbb{G}\|_2)\|\mathbb{A}_k^{-1}\|_2\|\nabla\Phi(x^k)\|_2$$
$$\le \frac{1 + \rho(1 + \theta)(1 - w_{min})}{\alpha\mu + (1 + \theta)(1 - w_{max})}\|\nabla\Phi(x^k)\|_2. \qquad\qquad \square$$

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

$$(25) \qquad\qquad \|\nabla\Phi(x) - \nabla\Phi(y)\|_2 \le \tilde{L}\|x - y\|_2, \ x, y \in \mathbb{R}^{np}.$$

Furthermore,

$$(26) \qquad\qquad \frac{\tilde{\mu}}{2}\|x - x^*\|_2^2 \le \Phi(x) - \Phi(x^*) \le \frac{1}{\tilde{\mu}}\|\nabla\Phi(x)\|_2^2$$

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

THEOREM 3.6. *Assume that the conditions of Theorem* 3.4 *are satisfied. Define*

$$(27) \qquad\qquad\qquad\qquad \varepsilon = \frac{\delta}{\beta^2\tilde{L}}$$

with β given by (22). *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, the Lipschitz property of $\nabla\Phi$, Theorem 3.4, and Lemma 3.5 yield

$$\Phi(x^{k+1}) - \Phi(x^*) = \Phi(x^k + \varepsilon s^k) - \Phi(x^*)$$
$$= \Phi(x^k) + \int_0^1 \nabla^T\Phi(x^k + t\varepsilon s^k)\varepsilon s^k dt - \Phi(x^*) \pm \varepsilon\nabla^T\Phi(x^k)s^k$$
$$\le \Phi(x^k) - \Phi(x^*) + \varepsilon\int_0^1 \|\nabla^T\Phi(x^k + t\varepsilon s^k) - \nabla^T\Phi(x^k)\|_2\|s^k\|_2 dt$$
$$+ \varepsilon\nabla^T\Phi(x^k)s^k$$

$$\leq \Phi(x^k) - \Phi(x^*) + \varepsilon \int_0^1 \tilde{L} t \varepsilon \|s^k\|_2^2 dt + \varepsilon \nabla^T \Phi(x^k) s^k$$

$$= \Phi(x^k) - \Phi(x^*) + \frac{1}{2} \varepsilon^2 \tilde{L} \|s^k\|_2^2 + \varepsilon \nabla^T \Phi(x^k) s^k$$

$$\leq \Phi(x^k) - \Phi(x^*) + \beta^2 \frac{\tilde{L}}{2} \varepsilon^2 \|\nabla \Phi(x^k)\|_2^2 - \varepsilon \delta \|\nabla \Phi(x^k)\|_2^2$$

(28)
$$= \Phi(x^k) - \Phi(x^*) + \left(\frac{\beta^2 \tilde{L}}{2} \varepsilon^2 - \varepsilon \delta \right) \|\nabla^2 \Phi(x^k)\|_2^2.$$

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

(29)
$$\phi(\varepsilon^*) = -\frac{\delta^2}{2\beta^2 \tilde{L}}.$$

Now, (28) and (29) give

$$\Phi(x^{k+1}) - \Phi(x^*) \leq \Phi(x^k) - \Phi(x^*) - \frac{\delta^2}{2\beta^2 \tilde{L}} \|\nabla \Phi(x^k)\|_2^2.$$

From (26), we also have

$$\Phi(x^k) - \Phi(x^*) \leq \frac{1}{\tilde{\mu}} \|\nabla \Phi(x^k)\|_2^2$$

and

$$-\frac{\delta^2}{2\beta^2 \tilde{L}} \|\nabla \Phi(x^k)\|_2^2 \leq - \left(\Phi(x^k) - \Phi(x^*) \right) \frac{\delta^2 \tilde{\mu}}{2\beta^2 \tilde{L}},$$

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,

$$\delta < \frac{1}{\alpha L + (1+\theta)(1 - w_{min})} \leq \frac{1}{\alpha \mu + (1+\theta)(1 - w_{max})}$$

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

and

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

We conclude with

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

and

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

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

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

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 3.6. However, local linear convergence can be obtained for the full step size, 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 3.7. *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 1 with*

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

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 (11) 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,

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

Moreover,

$$\|\mathbb{G}\mathbb{A}_k^{-1}\| = \max_j(\theta(1-w_{jj})\|(A_j^k)^{-1}\| + \sum_{i\in O_j} w_{ij}\|(A_j^k)^{-1}\|)$$

$$\le \max_j \frac{\theta(1-w_{jj}) + 1 - w_{jj}}{\alpha\mu + (1+\theta)(1-w_{jj})}.$$

Recalling (19) and the fact that the expression above is decreasing with respect to w_{jj}, we get

$$(30) \qquad \|\mathbb{G}\mathbb{A}_k^{-1}\| \le \frac{(1+\theta)(1-w_{min})}{\alpha\mu + (1+\theta)(1-w_{min})} =: \gamma,$$

and there holds $\|A_k\| \leq \alpha L + (1+\theta)(1-w_{min})$. Furthermore, (23), (24), and $\|\mathbb{L}_k\| \leq \rho$ imply

$$
\begin{aligned}
\|\mathbb{Q}_k\| &\leq \gamma + \gamma\rho(1+\theta)(1-w_{min}) + \gamma\rho(\alpha L + (1+\theta)(1-w_{min})) \\
&= \gamma + \rho\gamma(\alpha L + 2(1+\theta)(1-w_{min})) \\
&< \gamma + 1 - \gamma = 1.
\end{aligned}
$$
(31)

Thus, the statement is true with

$$
t = \gamma + \rho\gamma(\alpha L + 2(1+\theta)(1-w_{min})). \qquad \square
$$
(32)

Theorem 3.4 introduces an upper bound on the safeguard parameter ρ different from the one considered in Theorem 3.2. The relation between the two bounds depends on the choice of δ in Theorem 3.2. Taking a sufficiently small δ in Theorem 3.2, we obtain that ρ in Theorem 3.2 is larger. However, taking $\delta < \frac{1}{\alpha L + (1+\theta)(1-w_{min})}$ sufficiently close to $\frac{1}{\alpha L + (1+\theta)(1-w_{min})}$, ρ in Theorem 3.4 eventually becomes larger.

One way to interpret the relation between Theorems 3.2 and 3.3 on one hand and Theorem 3.4 on the other hand, as far as ρ is concerned, is as follows. Taking a very small δ, Theorem 3.3 allows for a quite large ρ but on the other hand it significantly decreases the admissible step size ε. At the same time, Theorem 3.4 corresponds in a sense to an opposite situation where ε is allowed to be quite large (in fact, equal to one), while ρ is quite restricted. Therefore, the two results exploit the allowed "degrees of freedom" in a different way.

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

THEOREM 3.8 (see [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 $\|y\|_* = \|\nabla^2\Phi(x^*)y\|$.

The two previous theorems imply the following corollary.

COROLLARY 3.9. *Assume that the conditions of Theorem* 3.7 *hold. Then there exists* $\eta > 0$ *such that for every* x^0 *satisfying* $\|x^0 - x^*\| \leq \eta$, *the sequence* $\{x^k\}$ *generated by algorithm DQN and* $\varepsilon = 1$ *converges linearly to* x^* *and*

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

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 3.10. *Assume that all loss functions* f_i *are strongly convex quadratic and that the conditions of Theorem* 3.7 *are satisfied. Let* $\{x^k\}$ *be a sequence generated by algorithm DQN with* $\varepsilon = 1$. *Then* $\lim_{k\to\infty} x^k = x^*$ *and*

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

for t *defined in Theorem* 3.7.

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

$$(33) \qquad \Phi(x) = \frac{1}{2}(x - x^*)^T \mathbb{B}(x - x^*),$$

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 3.7. 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

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

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 Theorems 3.4 and 3.7.

DQN-0. We first present the method DQN-0 which sets $\mathbb{L}_k = 0$. Clearly, for DQN-0, Theorems 3.2 and 3.4 hold, and thus we get linear convergence with the proper choice of ε and local linear convergence with $\varepsilon = 1$. The approximation of the Hessian inverse 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 3.4, 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 the ith 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 an additional vector u_i^0; see Algorithm 3). 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 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

$$(34) \qquad \nabla^2 \Phi(x^k) s_N^k + \nabla \Phi(x^k) = 0.$$

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

$$(35) \qquad M(\mathbb{L}_k) = \|\nabla^2 \Phi(x^k) s^k + \nabla \Phi(x^k)\|.$$

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,

$$(36) \qquad \nabla^2 \Phi(x^k) s^k + \nabla \Phi(x^k) = u^k + \nabla^2 \Phi(x^k) \mathbb{L}_k u^k,$$

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

$$(37) \qquad \mathbb{L}_k u^k = -(\nabla^2 \Phi(x^k))^{-1} u^k,$$

but (37) involves the inverse Hessian. Thus we approximate $(\nabla^2 \Phi(x^k))^{-1}$ by the Taylor expansion as follows. Clearly,

$$(38) \quad (\nabla^2 \Phi(x^k))^{-1} = (\alpha \nabla^2 F(x^k) + \mathbb{I} - \mathbb{Z})^{-1} = (\mathbb{I} - \mathbb{V}_k)^{-1}, \quad V_k = \mathbb{Z} - \alpha \nabla^2 F(x^k).$$

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,

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

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

$$(39) \qquad (\nabla^2 \Phi(x^k))^{-1} = (\mathbb{I} - \mathbb{V}_k)^{-1} \approx \mathbb{I} + \mathbb{V}_k$$

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

$$(40) \qquad \mathbb{L}_k u^k = -(\mathbb{I} + \mathbb{V}_k) u^k.$$

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 2, where step 6, the choice of $\mathbb{L}_k = \mathrm{diag}(\Lambda_1, \ldots, \Lambda_n)$, involves the steps presented below in Algorithm 3. Denote by u_i^k the ith $p \times 1$ block of u^k—the block which corresponds to node i.

Algorithm 3: Choosing $\mathbb{L}_k$ with DQN-2.
6.1. Each node i is calculated

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

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 Λ_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 Λ_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 safeguarding (step 6.4 in Algorithm 3) 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*

$$(41) \qquad \alpha < \min\left\{ \frac{1 + \lambda_n}{L}, \frac{w_{min}}{2L}, \frac{2\mu}{L^2} \right\}$$

and s^k is generated by (11) 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 (36) and (40) we obtain

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

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

$$(43) \qquad \|\mathbb{P}^k\| \leq \|\mathbb{Z}^2 - \alpha(\mathbb{Z}\nabla^2 F(x^k) + \nabla^2 F(x^k)\mathbb{Z})\| + \alpha^2 L^2 := \|U^k\| + \alpha^2 L^2.$$

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

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

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,

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

$$= \max_{j=1,\dots,n} \sum_{k=1}^{n} w_{kj} \sum_{i=1}^{n} w_{ik} - 2\alpha\mu \sum_{i=1}^{n} w_{ij}$$

$$= 1 - 2\alpha\mu.$$

So,

$$\text{(44)} \qquad \|\mathbb{P}^k\| \le h(\alpha),$$

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

$$\text{(45)} \qquad \|u^k\| \le \|\mathbb{G}\mathbb{A}_k^{-1}\| \|\nabla\Phi(x^k)\|.$$

Now,

$$\|\mathbb{G}\mathbb{A}_k^{-1}\| = \max_j (\theta(1 - w_{jj}) \|(A_j^k)^{-1}\| + \sum_{i \in O_j} w_{ij} \|(A_j^k)^{-1}\|)$$

$$\le \max_j \frac{\theta(1 - w_{jj}) + 1 - w_{jj}}{\alpha\mu + (1+\theta)(1 - w_{jj})}$$

$$= \frac{(1+\theta)(1 - w_{min})}{\alpha\mu + (1+\theta)(1 - w_{min})} < 1.$$

Therefore,

$$\text{(46)} \qquad \|u^k\| < \|\nabla\Phi(x^k)\|.$$

Putting together (42)–(46), for $\theta \ge 0$ and α satisfying (41) 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

$$\text{(47)} \qquad t = h(\alpha) = 1 - 2\alpha\mu + \alpha^2 L^2. \qquad \square$$

Applying Theorem 3.8 once again, we get the local linear convergence as stated in the following corollary.

COROLLARY 4.2. *Assume that the conditions of Theorem* 4.1 *hold. Then there exists η such that for every x^0 satisfying $\|x^0 - x^*\| \leq \eta$ the sequence $\{x^k\}$, generated by the DQN-2 method with steps* 6.1–6.3 *of Algorithm* 3 *and $\varepsilon = 1$, converges linearly to x^* and*

$$(48) \qquad \|\nabla^2 \Phi(x^*)(x^{k+1} - x^*)\| \leq t\|\nabla^2 \Phi(x^*)(x^k - x^*)\|, \quad k = 0, 1, \ldots,$$

holds with t given by (47).

We remark that for strongly convex quadratic f_i's, the result analogous to Theorem 3.10 holds in the sense of global linear convergence, i.e., inequality (48) holds for all k and arbitrary initial point x^0.

Remark 4.3. An interesting future research direction is to adapt and analyze convergence of the DQN methods in asynchronous environments, as it has been already numerically studied recently in [10]. Therein, it is shown that the studied second order methods still converge in an asynchronous setting, though with a lower convergence speed.

4.2. Discussion on the tuning parameters. Let us now comment on the choice of the involved parameters—matrix W and scalars $\alpha, \rho, \varepsilon, \theta$, and δ. 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 $0 < w_{min} \leq w_{max} < 1$. 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., [39].

We now discuss the choice of the parameters $\alpha, \rho, \theta, \varepsilon$, and δ. First, α defines the penalty reformulation (4), and therefore, it determines the asymptotic error that the algorithm achieves. The smaller the α, the smaller the limiting (saturation) error of the algorithm, but the slower the convergence rate. Thus, the parameter should be set a priori according to a given target accuracy; see also [40]. A practical guidance, as considered in [13], is to set $\alpha = 1/(\mathcal{K} L)$, where L is the Lipschitz gradient constant as in the paper, and $\mathcal{K} = 10\text{-}100$. Next, parameter $\theta \geq 0$ determines the splitting of the Hessian. It can simply be taken as $\theta = 0$, and a justification for this choice can be found in section 5. Next, the role of δ is mainly theoretical. Namely, Theorems 3.2 and 3.3 consider *generic* choices of $\mathbb{L}_k$'s, and they are worst-case type results. Therein, δ essentially trades off the guaranteed worst-case (global linear) convergence factor with the size of admissible range of the $\mathbb{L}_k$'s (size of the maximal allowed ρ). As per (18), a reasonable choice to balance the two effects is $\delta = \frac{1}{2(\alpha L + (1+\theta)w_{min})}$. Having set δ, the remaining two parameters, ε and ρ, can be set according to (27) and (18), respectively. As noted, the described choice of the triple (δ, ρ, ϵ) is a consequence of the worst case, conservative analysis with Theorems 3.2 and 3.3 (which still have a theoretical significance, though). In practice, we recommend setting $\delta = 0$, $\varepsilon = 1$, and ρ as the upper bound in (18) with $\delta = 0$.

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., [39], where

each node i needs to know only the degrees of its immediate neighbors. Such weight choice, as noted before, satisfies the imposed assumptions.

In order to set the scalar tuning parameters α, θ, ε, and ρ, each node i needs to know beforehand global quantities w_{min}, w_{max}, μ, and L. Each of these parameters represents either a maximum or a minimum of nodes' local quantities. For example, w_{max} is the maximum of the w_{ii}'s over $i = 1, \ldots, n$, 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 [34].

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 are considered: (1) strongly convex quadratic costs and (2) logistic (convex) loss functions. Very similar scenarios have been considered in [22, 23, 24]. 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 positive definite (symmetric matrix) and $a_i \in \mathbb{R}^p$ is a vector. Matrices B_i, $i = 1, \ldots, n$, are generated mutually independent, 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 $B_i = \widehat{Q}\mathrm{Diag}(\widehat{c}_i)\widehat{Q}^\top$, 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:

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

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

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

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

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

$f(x) = \sum_{i=1}^n f_i(x)$. 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 $b_{ij} = \mathrm{sign}\left((x_1^\star)^\top a_{ij} + x_0^\star + \varepsilon_{ij}\right)$,

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 [22, 23, 24], this weight choice coincides with that in [22, 23, 24].

The proposed methods DQN are compared with the methods NN-0, NN-1 and NN-2 proposed in [22]. The methods NN-ℓ with $\ell \geq 3$ are not numerically tested in [22] 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 (three vector communications per node).

With both the proposed methods and the methods in [22], the step size $\varepsilon = 1$ is used. Step size $\varepsilon = 1$ has also been used in [22, 23, 24]. 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 has 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 as the upper bound in (18) 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 error metric

$$\frac{1}{n} \sum_{i=1}^{n} \frac{\left\| x_i^k - x^\star \right\|_2}{\left\| x^\star \right\|_2}, \ x^\star \neq 0,$$

is used and referred 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 [22, 23, 24]. Also, the performance of DQN-0 and NN-1 practically matches. 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.

The improvements of DQN over NN are mainly due to the different splitting parameter $\theta = 0$. Actually, our numerical experience suggests that NN-ℓ may perform
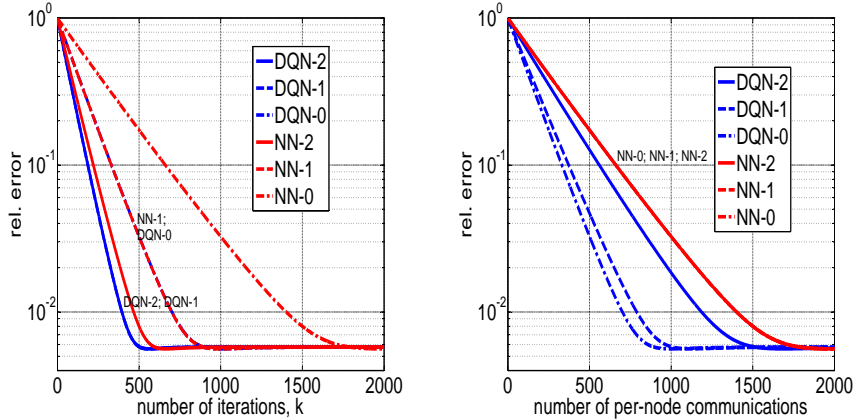
FIG. 1. *Relative error versus number of iterations k (left) and versus number of communications (right) for quadratic costs and n = 30-node network.*

better than DQN-ℓ with $\theta = 1$ for $\ell = 1, 2$. We provide an intuitive explanation for the advantages of choice $\theta = 0$, focusing on the comparison between NN-0 and DQN-0. Namely, the adopted descent directions with both of these methods correspond to the quality of the zeroth order Taylor expansion of the following matrix:

$$(49) \qquad \left(\mathbb{I} - (\mathbb{A}_k(\theta))^{-1} \mathbb{G} \right)^{-1} \approx \mathbb{I}.$$

In (49), with NN-0, we have $\mathbb{A}_k(\theta) = \mathbb{A}_k(\theta = 1)$, while with DQN-0, we have that $\mathbb{A}_k(\theta) = \mathbb{A}_k(\theta = 0)$; note that these two matrices are different. Now, the error (remainder) of the Taylor approximation is roughly of size $\|(\mathbb{A}_k(\theta))^{-1} \mathbb{G}\|$. In view of the upper bound in (30), we have with NN-0 that the remainder is of size

$$(50) \qquad \|(\mathbb{A}_k(1))^{-1} \mathbb{G}\| \approx 1 - \frac{\alpha \mu}{2(1 - w_{min})}$$

for small $\alpha \mu$. On the other hand, we have that the DQN-0's remainder is

$$\|(\mathbb{A}_k(0))^{-1} \mathbb{G}\| \approx 1 - \frac{\alpha \mu}{1 - w_{min}}.$$

Therefore, the remainder is (observed through this rough, but indicative, estimate) larger with NN-0, and that is why DQN-0 performs better. We can similarly compare NN-1 (which corresponds to the first order Taylor approximation of the matrix in (49)) with DQN-0. The remainder with NN-1 is roughly $\|(\mathbb{A}_k(1))^{-1} \mathbb{G}\|^2 \approx 1 - \frac{\alpha \mu}{1 - w_{min}}$, which equals the remainder estimate of DQN-0. This explains why the two methods perform very similarly. Finally, note that the upper bound on $\|\mathbb{G}\mathbb{A}_k^{-1}\|$ in (30) is an increasing function of $\theta \geq 0$ (the lower the θ, the better the bound), which justifies the choice $\theta = 0$ adopted here for DQN.

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 3 plots the relative error versus number of iterations (left) and number of per-node communications (right) for the logistic losses with variable dimension $p = 4$
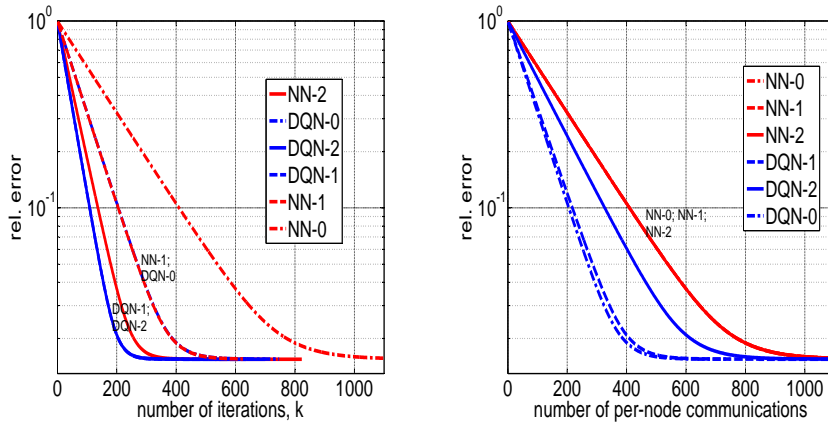
FIG. 2. *Relative error versus number of iterations k (left) and versus number of communications (right) for quadratic costs and n = 400-node network.*
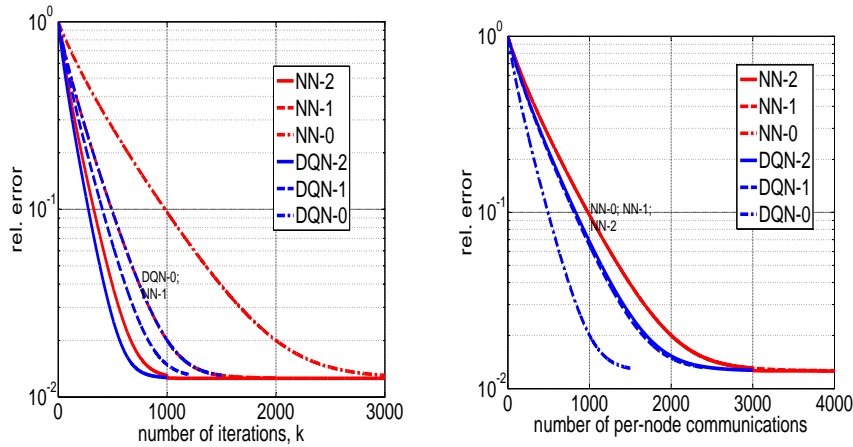

FIG. 3. *Relative error versus number of iterations k (left) and versus number of communications (right) for logistic costs and n = 30-node network.*

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. Extensions. As noted before, DQN methods converge not to the exact solution of (1) but to a solution neighborhood controlled by the step size α. As such, for the high solution accuracies required, they may not be competitive with distributed second order methods which converge to the exact solution [28, 38, 27].

However, we can exploit the results of [28] and "embed" the DQN algorithms in the framework of PMMs, just like [28] embeds the NN methods into the PMM
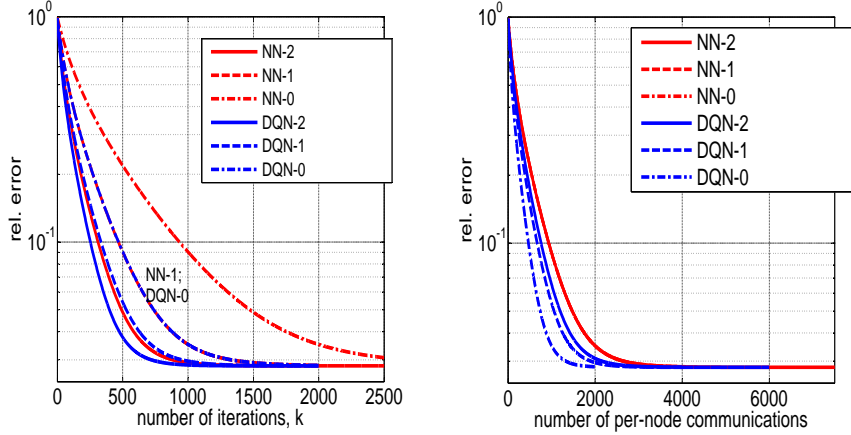
FIG. 4. *Relative error versus number of iterations k (left) and versus number of communications (right) for logistic costs and $n = 200$-node network.*

framework. We refer to the resulting algorithms as PMM-DQN-ℓ, $\ell = 0, 1, 2$. (Here, PMM-DQN-ℓ parallels DQN-ℓ in terms of complexity of approximating Hessians, and in terms of the communication cost per iteration.) It is worth noting that the contribution to embed distributed second order methods into the PMM framework is due to [28]. Here we extend [28] to demonstrate (by simulation) that the DQN-type Hessian approximations within the PMM framework yield efficient distributed second order methods.

We now briefly describe a general PMM-DQN method; for the methodology to devise distributed second order PMM methods, we refer to [28]. See also the appendix for further details. We denote by $\widehat{x}^k = \left(\widehat{x}_1^k, \ldots, \widehat{x}_n^k \right) \in \mathbb{R}^{np}$ the current iterate, where $\widehat{x}_i^k \in \mathbb{R}^p$ is node i's estimate of the solution to (1) at iteration k. Besides $\widehat{x}^k$ (the primal variable), the PMM-DQN method also maintains a dual variable $\widehat{q}^k = \left(\widehat{q}_1^k, \ldots, \widehat{q}_n^k \right) \in \mathbb{R}^{np}$, where $\widehat{q}_i^k \in \mathbb{R}^p$ is node i's dual variable at iteration k. Quantities $\widehat{\mathbb{H}}_k$, $\widehat{\mathbb{A}}_k$, $\widehat{\mathbb{G}}$, and $\widehat{g}_k$ defined below play a role in the PMM-DQN method and are, respectively, the counterparts of $\nabla^2 \Phi(x^k)$, $\mathbb{A}_k$, $\mathbb{G}$, and $\nabla \Phi(x^k)$ with DQN:

$$(51) \qquad \widehat{\mathbb{H}}_k = \nabla^2 F(\widehat{x}^k) + \beta \left(\mathbb{I} - \mathbb{Z} \right) + \epsilon_{\mathrm{pmm}} \mathbb{I} = \widehat{\mathbb{A}}_k - \widehat{\mathbb{G}},$$

$$(52) \qquad \widehat{\mathbb{A}}_k = \nabla^2 F(\widehat{x}^k) + \beta \left(\mathbb{I} - \mathbb{Z}_d \right) + \epsilon_{\mathrm{pmm}} \mathbb{I} + \beta \theta \left(\mathbb{I} - \mathbb{Z}_d \right),$$

$$(53) \qquad \widehat{\mathbb{G}} = \beta \mathbb{Z}_u + \beta \theta \left(\mathbb{I} - \mathbb{Z}_d \right),$$

$$(54) \qquad \widehat{g}_k = \nabla F(\widehat{x}^k) + \beta \left(\mathbb{I} - \mathbb{Z} \right) \widehat{x}^k + \widehat{q}^k.$$

Here, $\theta \geq 0$ is the splitting parameter as with DQN, $\beta > 0$ is the dual step size, and $\epsilon_{\mathrm{pmm}} > 0$ relates to the proximal term of the corresponding augmented Lagrangian; see [28] for details. We now present the general PMM-DQN algorithm. Note from step 2 in Algorithm 4 the analogous form of the Hessian inverse approximation as with DQN; the approximation is again parameterized with an $(np) \times (np)$ diagonal matrix $\widehat{\mathbb{L}}_k$.

Algorithm 4: PMM-DQN in vector format. Given $x^0 = 0$, $\beta, \epsilon_{\mathrm{pmm}}, \rho > 0$, $\theta \geq 0$. Set $k = 0$.

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

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

Step 2. Set

$$\widehat{s}^k = -(\mathbb{I} - \widehat{\mathbb{L}}_k\widehat{\mathbb{G}})\widehat{\mathbb{A}}_k^{-1}\,\widehat{g}_k.$$

Step 3. Set

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

Step 4. Set

$$\widehat{q}^{k+1} = \widehat{q}^k + (\mathbb{I} - \mathbb{Z})\,\widehat{x}^{k+1};\ k = k+1.$$

The same algorithm is presented below from the distributed implementation perspective. (Note that here we adopt notation similar to DQN, i.e., the (i,j)th $p \times p$ block of $\widehat{\mathbb{G}}$ is denoted by $\widehat{G}_{ij}$; the ith $p \times p$ diagonal block of $\widehat{\mathbb{L}}_k$ is denoted by $\widehat{\Lambda}_i^k$; and the ith $p \times p$ diagonal block of $\widehat{\mathbb{A}}_k$ is denoted by $\widehat{A}_i^k$.)

Algorithm 5: PMM-DQN—distributed implementation. At each node i, require $\beta, \rho, \epsilon_{\mathrm{pmm}} > 0$, $\theta \geq 0$.

1. Initialization: Each node i sets $k = 0$ and $\widehat{x}_i^0 = \widehat{q}_i^0 = 0$.
2. Each node i calculates

$$\widehat{d}_i^k = \left(\widehat{A}_i^k\right)^{-1}\left[\nabla f_i(\widehat{x}_i^k) + \beta \sum_{j \in O_i} w_{ij}\left(\widehat{x}_i^k - \widehat{x}_j^k\right) + \widehat{q}_i^k\right].$$

3. Each node i transmits $\widehat{d}_i^k$ to all its neighbors $j \in O_i$ and receives $\widehat{d}_j^k$ from all $j \in O_i$.
4. Each node i chooses a diagonal $p \times p$ matrix $\widehat{\Lambda}_i^k$ such that $\|\widehat{\Lambda}_i^k\| \leq \rho$.
5. Each node i calculates

$$\widehat{s}_i^k = -\widehat{d}_i^k + \widehat{\Lambda}_i^k \sum_{j \in \bar{O}_i} \widehat{G}_{ij}\,\widehat{d}_j^k.$$

6. Each node i updates its solution estimate as

$$\widehat{x}_i^{k+1} = \widehat{x}_i^k + \widehat{s}_i^k.$$

7. Each node i transmits $\widehat{x}_i^{k+1}$ to all its neighbors $j \in O_i$ and receives $\widehat{x}_j^{k+1}$ from all $j \in O_i$.
8. Each node i updates the dual variable $\widehat{q}_i^k$ as follows:

$$\widehat{q}_i^{k+1} = \widehat{q}_i^k + \sum_{j \in O_i} w_{ij}\left(\widehat{x}_i^{k+1} - \widehat{x}_j^{k+1}\right).$$

9. Set $k = k+1$ and go to step 2.

Note that, at step 2, each node i needs the neighbors' estimates $\widehat{x}_j^k$, $j \in O_i$. For $k \geq 1$, the availability of such information is ensured through step 7 of the previous iteration $k-1$; at $k = 0$, step 2 is also realizable as it is assumed that $\widehat{x}_i^0 = 0$ for all $i = 1, \ldots, n$. As with the DQN methods, different variants PMM-DQN-ℓ mutually differ in the choice of matrix $\widehat{\mathbb{L}}_k$. With PMM-DQN-0, we set $\widehat{\mathbb{L}}_k = 0$; with PMM-DQN-2, $\widehat{\mathbb{L}}_k$ is set as described below; with PMM-DQN-1, we set $\widehat{\mathbb{L}}_k = \widehat{\mathbb{L}} = \text{const}$ to $\widehat{\mathbb{L}}_0$, i.e., to the value of $\widehat{\mathbb{L}}_k$ from the first iteration of PMM-DQN-2.

We now detail how $\widehat{\mathbb{L}}_k$ is chosen with PMM-DQN-2. The methodology is completely analogous to DQN-2: the idea is to approximate the exact Newton direction $\widehat{s}_N^k$ which obeys the following Newton-type equation:

$$(55) \qquad\qquad \widehat{\mathbb{H}}_k\, \widehat{s}^k + \widehat{g}_k = 0$$

through Taylor expansions. Using (55) and completely analogous steps as with DQN-2, it follows that $\widehat{\mathbb{L}}_k$ is obtained through solving the following system of linear equations with respect to $\widehat{\mathbb{L}}_k$:

$$\widehat{\mathbb{L}}_k\, \widehat{u}^k = -\left(\frac{1}{\beta + \epsilon_{\text{pmm}}}\mathbb{I} + \frac{\beta}{(\beta + \epsilon_{\text{pmm}})^2}\mathbb{Z} - \frac{1}{(\beta + \epsilon_{\text{pmm}})^2}\nabla^2 F(\widehat{x}^k) \right) \widehat{u}^k,$$

$$(56) \quad \text{where} \quad \widehat{u}^k = \widehat{\mathbb{G}}\,\widehat{\mathbb{A}}_k^{-1}\, \widehat{g}_k.$$

The overall algorithm for setting $\widehat{\mathbb{L}}_k$ with PMM-DQN-2 (step 4 of Algorithm 5) is presented below.

Algorithm 6: Computing $\widehat{\mathbb{L}}_k$ with PMM-DQN-2.

4.1. Each node i calculates

$$\widehat{u}_i^k = \sum_{j \in \bar{O}_i} \widehat{G}_{ij}\, \widehat{d}_j^k.$$

4.2. Each node i transmits $\widehat{u}_i^k$ to all its neighbors $j \in O_i$ and receives $\widehat{u}_j^k$ from all $j \in O_i$.

4.3. 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 $\widehat{\Lambda}_i$):

$$\widehat{\Lambda}_i\, \widehat{u}_i^k = -\left[\left(\frac{1}{\beta + \epsilon_{\text{pmm}}} + \frac{\beta\, w_{ii}}{(\beta + \epsilon_{\text{pmm}})^2} \right) I - \frac{1}{(\beta + \epsilon_{\text{pmm}})^2}\nabla^2 f_i(\widehat{x}_i^k) \right]$$

$$\times \widehat{u}_i^k - \frac{\beta}{(\beta + \epsilon_{\text{pmm}})^2} \sum_{j \in O_i} w_{ij}\, \widehat{u}_j^k.$$

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

Simulations. We now compare by simulation the PMM-DQN-ℓ methods with the ESOM-ℓ algorithms proposed in [28], the DQM algorithm in [27], and different variants of the Newton Raphson consensus (NRC) algorithm proposed in [38].

The simulation setup is as follows. The network is an instance of the random geometric graph with $n = 30$ nodes and 166 links. The optimization variable dimension is $p = 4$, and the local nodes costs are strongly convex quadratic, generated at random in the same way as with the quadratic costs examples in section 5. With all methods which involve weighted averaging (ESOM-ℓ, PMM-DQN-ℓ, and NRC), we use the Metropolis weights. We consider the ESOM-ℓ and PMM-DQN-ℓ methods for $\ell = 0, 1, 2$. With the methods ESOM-ℓ and PMM-DQN-ℓ, we set the proximal constant $\epsilon_{\text{pmm}} = 10$ (see [28] for details). Further, we tune the dual step size β separately for each of these methods to the best by considering the following candidate values: $\beta \in \{10^{-4}, 10^{-3.5}, 10^{-3}, \ldots, 10^{3.5}, 10^4\}$, i.e., a grid of points equidistant on the $\log_{10}$ scale with the half-decade spacing. The algorithm DQM has the tuning step size parameter $c > 0$ (see [27] for details) which we also tune to the best using the same grid of candidate values. Regarding the methods proposed in [38], we consider both the standard (NRC) and the accelerated (FNRC) algorithm variant. These methods

have a communication cost per node, per iteration which is quadratic in p, due to exchanging local Hessian estimates. (Specifically, as it is sufficient to transmit the upper-triangular part of a local Hessian (due to the matrix symmetry), each node per iteration transmits $p \times (p+1)/2$ scalars for the Hessian exchange.) This is different from the ESOM, DQM, and PMM-DQN methods which have a cost linear in p. We also consider the Jacobi variants in [38] (both the standard, JC, and the accelerated, FJC, variant) which approximate local Hessians through diagonal matrices and hence their communication cost per node, per iteration reduces to a cost linear in p. With NRC, FNRC, JC, and FJC, we set their step size ϵ to unity (see [38] for details), as this (maximal possible) step size value yielded fastest convergence. We observed that JC and FJC converged with a nonzero limiting error, while decreasing the value of ϵ did not improve the limiting error of the method while slowing down the methods. Hence, with all the methods considered, we tune their step sizes to the best (up to the finite candidate grid points resolution). With FNRC and FJC, we set the acceleration parameter ϕ in the same way as in [38] (see p. 10 of [38].) With all PMM-DQN methods, we do not use safeguarding ($\rho = +\infty$), and we set $\theta = 0$. With all the methods considered, the primal variables—solution estimates (and dual variables, if they exist)—are initialized with zero vectors. The error metric is the same as with the quadratic example in section 5.

Figure 5 compares the PMM-DQN-ℓ algorithms and the ESOM-ℓ algorithms in [28] in terms of the number of iterations (left) and the number of per-node communications (right). We can see that, in terms of iterations, for each fixed ℓ, the corresponding PMM-DQN method performs better than the corresponding ESOM method. The exception is the case $\ell = 0$, where the two methods are comparable. The same conclusions hold for the number of communications also. Further, in terms of the number of iterations, PMM-DQN-1 and PMM-DQN-2 are the best among all methods; in terms of communications, PMM-DQN-1 is the best method among all PMM-DQN and ESOM method considered.

In Figure 6, we compare the best among the PMM-DQN-ℓ methods, the best among the ESOM-ℓ methods (in terms of iterations, the best are PMM-DQN-1 and
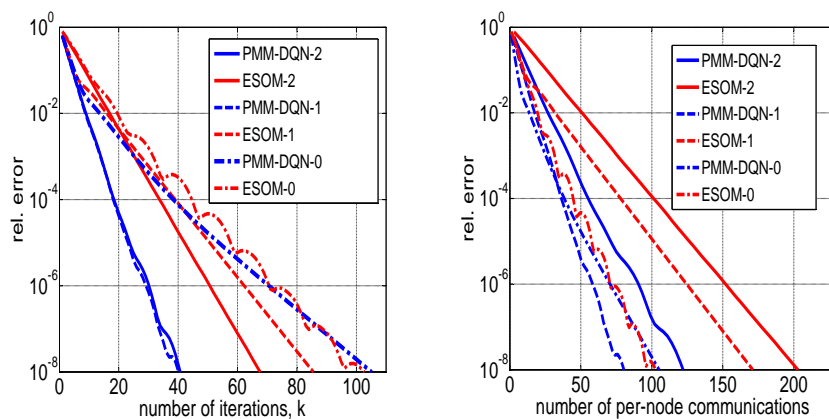

FIG. 5. *Comparison between the PMM-DQN-ℓ algorithms and the ESOM-ℓ algorithms in* [28]. *The figures plot relative error versus number of iterations k (left) and versus number of communications (right) for quadratic costs and $n = 30$-node network.*
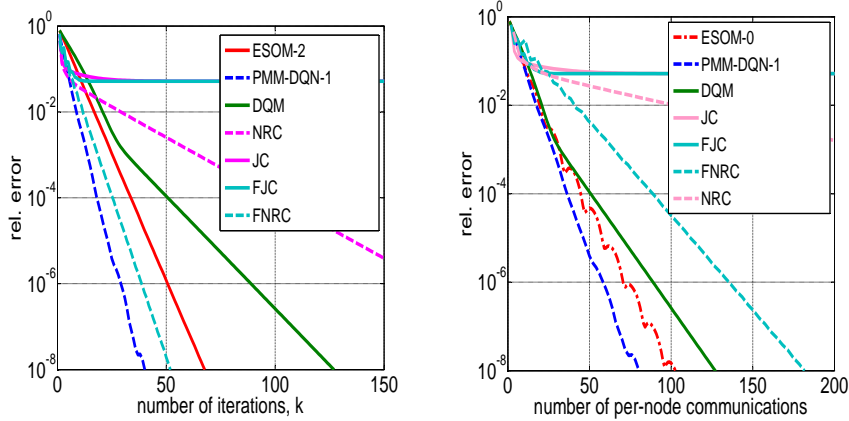
FIG. 6. *Comparison between the DQM algorithm in* [27], *the NRC algorithm in* [38], *the best among the PMM-DQN-ℓ algorithms, and the best among the ESOM-ℓ algorithms. The figures plot relative error versus number of iterations k (left) and versus number of communications (right) for quadratic costs and n = 30-node network.*

ESOM-2, while in terms of communications, the best are PMM-DQN-1 and ESOM-0), DQM, and the NRC methods group (NRC, FNRC, JC, and FJC). We can see that the PMM-DQN-1 method converges faster than all other methods, in terms of both iterations and communications.

7. Conclusions. The problem under consideration is defined by an aggregate, networkwide sum cost function across nodes in a connected network. It is assumed that the cost functions are convex and differentiable, while the network is characterized by a symmetric, stochastic matrix W that fulfills standard assumptions. The proposed methods are designed by exploiting a 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 $\mathbb{L}_k$. The key point in the proposed class of methods is to exploit the structure of Hessian and replace the dense part of the inverse Hessian by an inexpensive linear approximation, determined by matrix $\mathbb{L}_k$. Depending on the choice of $\mathbb{L}_k$, one can define different methods, and three of such choices are analyzed in this work. 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.

The three specific methods are analyzed in detail, termed DQN-0, DQN-1, and DQN-2. They are defined by the three different choices of matrix $\mathbb{L}_k$—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 inexpensive distributed implementation. For the last choice of the time-varying matrix, we have shown local linear convergence for the full step size without safeguarding.

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 favorably with the benchmark methods. Finally, applying the recent contributions of [28], the proposed distributed second order methods were extended to the framework of PMM. Unlike DQN, the modified methods converge to the exact solution and further enhance the performance when high solution accuracies are required.

Appendix. Following [28], we briefly explain how we derive the PMM-DQN methods. The starting point for PMM-DQN is the quadratically approximated PMM method, which takes the following form (see [28] for details and derivations):

$$(57) \qquad \widehat{x}^{k+1} = \widehat{x}^k - \widehat{\mathbb{H}}_k^{-1} \left(\nabla F(\widehat{x}^k) + \widehat{q}^k + \beta(\mathbb{I} - \mathbb{Z})\widehat{x}^k \right),$$

$$(58) \qquad \widehat{q}^{k+1} = \widehat{q}^k + \beta(\mathbb{I} - \mathbb{Z})\widehat{x}^{k+1}.$$

Here, $\beta > 0$ is the (dual) step size, $\widehat{\mathbb{H}}_k$ is given in (51), and $\widehat{x}^k \in \mathbb{R}^{np}$ and $\widehat{q}^k \in \mathbb{R}^{np}$ are respectively the primal and dual variables at iteration $k = 0, 1, \ldots$, initialized by $\widehat{x}^0 = \widehat{q}^0 = 0$.

The challenge for distributed implementation of (57)–(58) is that the inverse of $\widehat{\mathbb{H}}_k$ does not respect the sparsity pattern of the network. The ESOM methods, proposed in [28], approximate the inverse of $\widehat{\mathbb{H}}_k$ following the NN-type approximations [22]. Here, we extend such possibilities and approximate the inverse of $\widehat{\mathbb{H}}_k$ through the DQN-type approximations. This is defined in (51)–(53) and algorithm PMM-DQN in section 6.

As noted, the matrix $\mathbb{L}_k = 0$ with PMM-DQN-0, and it is $\mathbb{L}_k = \mathbb{L}_0 = \text{const}$ with PMM-DQN-1, where $\mathbb{L}_0$ is the matrix from the first iteration of the DQN-2 method. It remains to derive $\mathbb{L}_k$ for PMM-DQN-2, as given in section 6. As noted in section 6, we approximate the Newton equation in (55). The derivation steps are the same as with DQN-2, with the following identification: $\nabla^2\Phi(x^k)$ in (34) is replaced with $\widehat{\mathbb{H}}_k$ in (55), and $\nabla\Phi(x^k)$ with $\widehat{g}^k$ in (55). Then, (37) with DQN-2 transforms into the following equation with PMM-DQN-2:

$$(59) \qquad \widehat{\mathbb{L}}_k \widehat{u}^k = -\widehat{\mathbb{H}}_k^{-1}\widehat{u}^k,$$

where $\widehat{u}^k$ is given in (56). Finally, it remains to approximate $\widehat{\mathbb{H}}_k^{-1}$ through a first order Taylor approximation, as follows:

$$\widehat{\mathbb{H}}_k^{-1} = \left[(\beta + \epsilon_{\text{pmm}}) \left(\mathbb{I} - \left(\frac{\beta}{\beta + \epsilon_{\text{pmm}}}\mathbb{Z} - \frac{1}{\beta + \epsilon_{\text{pmm}}}\nabla^2 F(\widehat{x}^k) \right) \right) \right]^{-1}$$

$$\approx \frac{1}{\beta + \epsilon_{\text{pmm}}} \left[\mathbb{I} + \frac{\beta}{\beta + \epsilon_{\text{pmm}}}\mathbb{Z} - \frac{1}{\beta + \epsilon_{\text{pmm}}}\nabla^2 F(\widehat{x}^k) \right].$$

The above Taylor approximation is well defined if the spectral radius of matrix $(\frac{\beta}{\beta+\epsilon_{\text{pmm}}}\mathbb{Z} - \frac{1}{\beta+\epsilon_{\text{pmm}}}\nabla^2 F(\widehat{x}^k))$ is strictly less than one. It is easy to verify that this will be the case if the (positive) parameters β and ϵ_{pmm} satisfy $\beta > \frac{1}{2}\max\{0, L - \epsilon_{\text{pmm}}\}$.

REFERENCES

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

[2] R. H. Byrd, G. M. Chin, W. Neveitt, and J. Nocedal, *On the use of stochastic Hessian information in optimization methods for machine learning*, SIAM J. Optim., 21 (2011), pp. 977–995.

[3] R. H. Byrd, G. M. Chin, J. Nocedal, and Y. Wu, *Sample size selection in optimization methods for machine learning*, Math. Program., 134 (2012), pp. 127–155.

[4] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, *A stochastic quasi-Newton method for large-scale optimization*, SIAM J. Optim., 26 (2016), pp. 1008–1031.

[5] F. Cattivelli and A. H. Sayed, *Diffusion LMS strategies for distributed estimation*, IEEE Trans. on Signal Process., 58 (2010), pp. 1035–1048.

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

[7] A. Daneshmand, F. Facchinei, V. Kungurtsev, and G. Scutari, *Hybrid random/deterministic parallel algorithms for nonconvex big data optimization*, IEEE Trans. Signal Process., 63 (2014), pp. 3914–3929.

[8] R. S. Dembo, S. C. Eisenstat, and T. Steinhaug, *Inexact Newton method*, SIAM J. Numer. Anal., 19 (1982), pp. 400–409.

[9] J. R. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics in Appl. Math., SIAM, Philadelphia, 1996.

[10] M. Eisen, A. Mokhtari, and A. Ribeiro, *A Decentralized Quasi-Newton Method for Dual Formulations of Consensus Optimization*, in Proceedings of the IEEE Conference on Decision and Control, 2016.

[11] D. G. Feingold and R. S. Varga, *Block diagonally dominant matrices and generalizations of the Gershgorin circle theorem*, Pacific J. Math., 12 (1962), pp. 1241–1250.

[12] M. P. Friedlander and M. Schmidt, *Hybrid deterministic-stochastic methods for data fitting*, SIAM J. Sci. Comput., 34 (2012), pp. 1380–1405.

[13] D. Jakovetić, D. Bajović, N. Krejić, and N. Krklec Jerinkić, *Distributed gradient methods with variable number of working nodes*, IEEE Trans. Signal Process., 64 (2016), pp. 4080–4095.

[14] D. Jakovetić, J. Xavier, and J. M. F. Moura, *Fast distributed gradient methods*, IEEE Trans. Automat. Control, 59 (2014), pp. 1131–1146.

[15] D. Jakovetić, J. M. F. Moura, and J. Xavier, *Distributed Nesterov-like gradient algorithms*, in Proceedings of the 51st IEEE Conference on Decision and Control, Maui, HI, 2012, pp. 5459–5464.

[16] S. Kar, J. M. F. Moura, and K. Ramanan, *Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication*, IEEE Trans. Inform. Theory, 58 (2012), pp. 3575–3605.

[17] N. Krejić and N. Krklec, *Variable sample size methods for unconstrained optimization*, J. Comput. Appl. Math., 245 (2013), pp. 213–231.

[18] N. Krejić and N. Krklec Jerinkić, *Nonmonotone line search methods with variable sample size*, Numer. Algorithms 68 (2015), pp. 711–739.

[19] N. Krejić and Z. Lužanin, *Newton-like method with modification of the right-hand side vector*, Math. Comp., 71 (2002), pp. 237–250.

[20] I. Lobel, A. Ozdaglar, and D. Feijer, *Distributed multi-agent optimization with state-dependent communication*, Math. Program., 129 (2014), pp. 255–284.

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

[22] A. Mokhtari, Q. Ling, and A. Ribeiro, *An approximate Newton method for distributed optimization*, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Brisbane, Australia, 2015, pp. 2959–2963.

[23] A. Mokhtari, Q. Ling, and A. Ribeiro, *Network Newton—Part* I: *Algorithm and Convergence*, arXiv:1504.06017, 2015.

[24] A. Mokhtari, Q. Ling, and A. Ribeiro, *Network Newton–Part* II: *Convergence Rate and Implementation*, arXiv:1504.06020, 2015.

[25] A. MOKHTARI AND A. RIBEIRO, *Regularized stochastic BFGS method*, IEEE Trans. Signal Process., 62 (2014), pp. 6089–6104.

[26] A. MOKHTARI AND A. RIBEIRO, *Global convergence of online limited memory BFGS method*, J. Mach. Learn. Res., 17 (2016), pp. 1–35.

[27] A. MOKHTARI, W. SHI, Q. LING, AND A. RIBEIRO, *DQM: Decentralized quadratically approximated alternating direction method of multipliers*, IEEE Trans. Signal. Process., 64 (2016), pp. 5158–5173.

[28] A. MOKHTARI, W. SHI, Q. LING, AND A. RIBEIRO, *A decentralized second order method with exact linear convergence rate for consensus optimization*, IEEE Trans. Signal Inform. Process. Netw., 2 (2016), pp. 507–522.

[29] J. MOTA, J. XAVIER, P. AGUIAR, AND M. PÜSCHEL, *Distributed optimization with local domains: Applications in MPC and network flows*, IEEE Trans. Automat. Control, 60 (2015), pp. 2004–2009.

[30] A. NEDIĆ AND A. OZDAGLAR, *Distributed subgradient methods for multi-agent optimization*, IEEE Trans. Automat. Control, 54 (2009), pp. 48–61.

[31] S. RAM AND A. NEDIĆ, AND V. VEERAVALLI, *Distributed stochastic subgradient projection algorithms for convex optimization*, J. Optim. Theory Appl., 147 (2011), pp. 516–545.

[32] S. S. RAM, A. NEDIĆ, AND V. VEERAVALLI, *Asynchronous gossip algorithms for stochastic optimization*, in Proceedings of the 48th IEEE International Conference on Decision and Control, Shanghai, China, 2009, pp. 3581–3586.

[33] I. D. SCHIZAS, A. RIBEIRO, AND G. B. GIANNAKIS, *Consensus in ad hoc WSNs with noisy links – Part I: Distributed estimation of deterministic signals*, IEEE Trans. Signal Process., 56 (2009), pp. 350–364.

[34] G. SHI AND K. H. JOHANSSON, *Finite-Time and Asymptotic Convergence of Distributed Averaging and Maximizing Algorithms*, arXiv:1205.1733, 2012.

[35] W. SHI, Q. LING, K. YUAN, G. WU, AND W. YIN, *On the linear convergence of the ADMM in decentralized consensus optimization*, IEEE Trans. Signal Process., 62, 2013.

[36] W. SHI, Q. LING, G. WU, AND W. YIN, *EXTRA: An exact first-order algorithm for decentralized consensus optimization*, SIAM J. Optim., 25 (2015), pp. 944–966.

[37] E. WEI, A. OZDAGLAR, AND A. JADBABAIE, *A distributed Newton method for network utility maximization* I: *Algorithm*, IEEE Trans. Automat. Control, 58 (2013), pp. 2162–2175.

[38] D. VARAGNOLO, F. ZANELLA, A. CENEDESE, G. PILLONETTO, AND L. SCHENATO, *Newton-Raphson consensus for distributed Convex optimization*, IEEE Trans. Automat. Control, 61, 2016.

[39] L. XIAO, S. BOYD, AND S. LALL, *A scheme for robust distributed sensor fusion based on average consensus*, in Proceedings of Information Processing in Sensor Networks, Los Angeles, CA, 2005, pp. 63–70.

[40] L. YUAN, Q. LING, AND W. YIN, *On the Convergence of Decentralized Gradient Descent*, arXiv:1310.7063, 2013.

[41] M. ZARGHAM, A. RIBEIRO, AND A. JADBABAIE, *Accelerated dual descent for constrained convex network flow optimization*, in Proceedings of the IEEE 52nd Annual Conference on Decision and Control, 2013, pp. 1037–1042.