Received: 27 January 2019 | Revised: 25 July 2019 | Accepted: 11 October 2019

DOI: 10.1002/asjc.2284

REGULAR PAPER

WILEY

Distributed convex optimization based on ADMM and belief propagation methods

Wenlong Ma[1] | **Huanshui Zhang**[1] | **Minyue Fu**[2,3]

[1]School of Control Science and Engineering, Shandong University, Jinan, 250061, China

[2]School of Electrical Engineering and Computing, University of Newcastle, 2308, NSW, Australia

[3]School of Automation, Guangdong Unversity of Technology, Guangzhou, China

Correspondence
Minyue Fu, School of Electrical Engineering and Computing, University of Newcastle, NSW 2308 Australia.
Email: minyue.fu@newcastle.edu.au

Funding information
National Natural Science Foundation of China, Grant/Award Number: 61633014, 61573221 and U1701264

Abstract

We consider a distributed convex optimization problem in which a connected network of agents collaboratively seeks to minimize the sum of their local objective functions over a common decision variable. We propose a new distributed optimization method in the Alternating Direction Method of Multipliers (ADMM) framework, But our method combines the celebrated Belief Propagation(BP) algorithm and relaxation iteration method to achieve distributed optimization. Numerical simulation shows that our proposed algorithms have good convergence speed. We also discuss the trade-off between the convergence rate and the required communication and computational complexities.

KEYWORDS

alternating direction method of multipliers, belief propagation, distributed optimization, relaxation iteration method

1 | INTRODUCTION

In recent years, distributed convex optimization methods for solving convex optimization problems over networks have received considerable interest. In a connected, undirected network with N nodes, each of which having a local private convex cost function $f_i : R^n \to R, i \in 1, 2, \ldots, N$, we focus on iterative, distributed algorithms that solve the convex optimization problem:

$$\min_{x \in R^n} f(x) := \sum_{i=1}^{N} f_i(x), \tag{1}$$

where $f(x)$ is the aggregate cost function and $x \in R^n$ is a global decision variable to be optimized. This problem has found various applications in multi-agent control [1,2], sensor fusion in wireless sensor networks [3], distributed learning [4] to just name a few.

In a distributed sense, the main problem can also be rewritten by introducing a local copy $x_i \in R^n$ of the global variable x for each agent i, that is, (1) becomes

$$\min_{\mathbf{x} \in R^{nN}} F(\mathbf{x}) := \sum_{i=1}^{N} f_i(x_i), \text{ s. t. } x_i = x_j \quad \forall i, j, \tag{2}$$

where $\mathbf{x} = \text{col}\{x_1, x_2, \ldots, x_N\}$.

Distributed methods for solving (2) can be divided into two classes: gradient descent methods and Lagrangian dual methods. In the gradient descent methods framework, the pioneering work is distributed subgradient methods (DSM) in the literature [5] which is simple to implement, but the convergence speed is slow. The work [6] proposes an accelerated distributed Nesterov gradient method with multi-step consensus inner iterations and achieves rates $O(1/k^2)$ for convex and smooth cost functions. Here k is the number of gradient evaluation iterations. [7] further shows the accelerated distributed Nesterov gradient method can achieve a rate $O((1 - \sqrt{\frac{\mu}{\sigma}})^2)$ for μ-strongly convex and σ-smooth functions. Further

utilizing smoothness, [8,9] has a linear convergence rate for strongly convex functions and the key step in the algorithm is a novel gradient estimation scheme that involves historical gradient information. The paper [10] proposes a novel decentralized exact first-order algorithm (abbreviated as EXTRA) with global linear converge to the exact solution with a constant step size for cost functions with strong convexity and smoothness, and [11] further analyzes the equivalence between the EXTRA and the augmented Lagrangian dual method. Based on the conditions of strong convexity and smoothness, further using the Newton steps' Taylor expansion, Network Newton (NN) method in [12] achieves an at least linear convergence rate. In the Lagrangian dual framework, the widely known method is the Alternating Direction Method of Multipliers (ADMM) [13]. Under the distributed ADMM iteration framework, each agent is required to repeatedly solve an specific sub-problem to global optimality which is computationally expensive. Many papers, such as [14,15], which are mainly based on the gradient descent method above, need to use a large number of inner iterations to achieve consensus of local variables to achieve iterative updates. Although these works [14–18] show that the distributed ADMM can exhibit a linear convergence rate for objective functions with strong convexity and smoothness, they require a large amount of calculation and information exchange as the cost.

In this paper, we assume that all local objective functions are twice continuously differentiable, σ-smooth and strict convexity. This assumption ensures that the Hessian matrix for each cost function is positive definite and bounded. We extend [11,15] and propose a different approach of distributed optimization in the ADMM framework for solving (2). Our method is designed to exploit the special Lagrangian structures [15], for the series of optimization problems in the primary iterations, with a different method of gradient descent, we extract the optimality conditions of this series of optimization problems to form a high-dimensional linear equations, then we solve this linear equation system in a distributed way to complete the whole iterative process. In the specific implementation, based on Gaussian belief propagation [19–21] and finite-time consensus protocol [22–25], we propose our Generalized Belief Propagation algorithm. In order to compare the effect of this algorithm, based on the classic relaxation methods [26–28], we generalize the Generalized Gradient method for comparison.

The contribution of the article is that we propose a new distributed optimization method with linear convergence rate. Our method is to reconstruct the generalized gradient direction by solving a set of linear equations, which is different from the accelerated gradient descent method and consensus-based ADMM. In addition, our algorithm can also use the second order gradient information directly to accelerate convergence, which is completely different from Taylor extension in [12]. Number simulation shows that our algorithm requires only a small number of internal iterations to yield linear convergence. Our methods require less information exchange, calculation and storage relative to the existing (accelerated) gradient descent method($1/k^2$ in [6]) with the same moderate precision requirements.

The problem we are considering is the basic problem that has been extensively studied in many more complex scenarios and has resulted in a series of discrete time and continuous time algorithms. In this paper, we propose another way to deal with this simple basic problem that is not weaker than the conventional method that has emerged. The next step is to further attenuate the problem's assumptions and study the problem in non-smooth and convex cases.

The rest of the paper is structured as follows: Section II states the distributed convex optimization problem and our network model. Section III presents our Generalized Belief Propagation algorithm and the Generalized Gradient algorithm. Section IV demonstrates the performance of the proposed algorithm. Section V gives the summary.

Basic Notations: For a matrix $A \in R^{m\times m}$, we denote its transpose matrix by A^T. Let $\mathbf{1}_n$ denote the vector of n ones, and $\mathbf{I}_n$ the $n \times n$ identity matrix. We denote the standard Euclidean norm of vector $\mathbf{x} \in R^n$ by $\|\mathbf{x}\|$. Diag$(\cdot)$ is a diagonal matrix. A symmetric matrix $A \in R^{n\times n}$ is called (strictly) diagonally dominant if $|A_{ii}|(>) \geqslant \sum_{i\neq j}|A_{ij}|$ for all $i = 1, 2, \ldots n$.

2 | PROBLEM FORMULATION

This section formalizes the distributed optimization problem in the network model.

2.1 | Network Models and Assumptions

We consider a (sparse) network, characterized by a triplet $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ which is an undirected connected simple graph with N nodes (or agents) and M edges, where $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ denotes the set of undirected edges. $\mathcal{N}_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$ is the neighborhood of agent i.

We assume $i < j$ in the representation of the edge e_{ij}. The edge-node incidence matrix $C \in R^{M\times N}$ is such that each column represents an agent and each row corresponds to an edge. In the column of this matrix, all edges are sorted according to dictionary ordering and the elements of C is

defined as:

$$C_{((ij),s)} = \begin{cases} 1 & \text{edge } e_{ij} \in \mathcal{E} \text{ and } s = i, \\ -1 & \text{edge } e_{ij} \in \mathcal{E} \text{ and } s = j, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$A \in R^{N\times N}$ is the symmetric adjacency matrix with 0 on the diagonal and 1 in the (i,j)th position if node i is connected to node j. The graph Laplacian matrix of $\mathcal{G}$, $L := C^TC = D - A$, is a positive semi-definite matrix and satisfies: $L\mathbf{1} = \mathbf{0}$. Here the diagonal matrix D is the degree matrix of $\mathcal{G}$.

Assumption 1. Graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is connected, undirected, and simple (no self-loops or multiple edges between two nodes).

Assumption 1 is standard for distributed optimization and it implies that any two agents in the network can always influence each other in the long run.

2.2 | Problem formulation

In the network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ of N agents, $f_i : R^n \mapsto R$ is a proper closed convex objective function known only by agent i. Our aim is to compute the minimizer $x^\star$ of $f(x)$ through the cooperation of the agents in the network, that is, to solve the problem (1) or the equivalent problem (2).

Throughout this paper, we assume that the set of minimizers of f is nonempty. Combining the strict convexity of the function (Assumption 2), the optimal value point is unique and we represent it as x^*.

Definition 1 (Strict Convexity)**.** A functions ξ are strictly convex, i.e.,

$$\xi(y) > \xi(x) + \langle \nabla\xi(x), y - x\rangle, \tag{4}$$

hold for all $x; y \in R^n$.

Definition 2 (σ-Smooth)**.** A function ξ is σ-smooth with $\sigma > 0$ if it is continuously differentiable and $\sigma/2\|\cdot\|^2 - \xi$ is convex, or equivalently that

$$\xi(y) \le \xi(x) + \langle \nabla\xi(x), y - x\rangle + \frac{\sigma}{2}\|x - y\|^2,$$

holds for all $x; y \in R^n$.

Assumption 2. $\forall i \in \mathcal{V}, f_i$ is twice continuously differentiable, σ_i-smooth and strict convexity.

Lemma 1 (Bounded Hessian)**.** *Under Assumption 2, $\forall i \in \mathcal{V}$, the Hessian matrix of every f_i is bounded, i.e., $0 < \nabla^2 f_i(x) \le \sigma_i$.*

Lemma 1 is an obvious result under Assumption 2. In order to concisely express and highlight the main ideas, we limit $x_i \in R$ to study the problem. The the following results can be easily scaled to multi-dimensional situations.

3 | MAIN RESULTS

In this section, we first place the problem 5 in the framework of ADMM, and then we show the algorithm using relaxation iterative as a comparison with our proposed algorithm. Finally, we present our algorithm and related proofs.

3.1 | Distributed ADMM optimization

The connectivity of the graph ensures that the problem (2) can also be rewritten as

$$\begin{aligned} &\min_{x_i\in R, i=1,2,\dots,N} F(\mathbf{x}) := \sum_{i=1}^{N} f_i(x_i), \\ &\text{s.t.} \quad x_i - x_j = 0, \quad \forall j \in \mathcal{N}_i. \end{aligned} \tag{5}$$

where $\mathbf{x} = \text{col}\{x_1, x_2, \dots, x_N\} \in R^{n*N}$. Using the incidence matrix C in (3), we get a compact form of the main optimization problem as

$$\begin{aligned} &\mathbf{x}^* = \arg \min_{\mathbf{x}\in R^N} F(\mathbf{x}), \\ &s.t. \quad C\mathbf{x} = 0, \end{aligned} \tag{6}$$

where $\mathbf{x}^* = \text{col}\{x^*, x^*, \dots, x^*\} \in R^{n*N}$. In order to solve the problem (6), we use a special augmented Lagrangian: $\mathcal{L}_\rho : R^N \times R^M \to R$:

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{y}) = F(\mathbf{x}) + \mathbf{y}^T C\mathbf{x} + \frac{\rho}{2}\mathbf{x}^T L\mathbf{x}, \tag{7}$$

where ρ is a position constant, and the dual variable $\mathbf{y} \in R^M$ is the Lagrange multiplier. As $L = C^TC$, the augmentation term $\frac{\rho}{2}\mathbf{x}^T L\mathbf{x}$ is null when the variable $\mathbf{x}$ is a feasible solution of (6). Each element of $\mathbf{y}$ corresponds to an edge of the connected graph $\mathcal{G}$ and the order of the edges is the same as the order of the corresponding edges in matrix C.

In the framework of ADMM, the primal variables $\mathbf{x}$ and the Lagrange multiplier $\mathbf{y}$ can be iterated as follows: starting from some initial vector $(\mathbf{x}(0), \mathbf{y}(0))$, at iteration $k > 0$, the variables are updated as

$$\mathbf{x}(k+1) = \arg \min_{\mathbf{x}} \mathcal{L}_\rho(\mathbf{x}, \mathbf{y}(k)), \tag{8}$$

$$\mathbf{y}(k+1) = \mathbf{y}(k) + \rho C\mathbf{x}(k+1). \tag{9}$$

For the distributed implementation of the ADMM iteration, the specific iterative calculation of (9) in every $y_{(ij)}$ (corresponding to the edge e_{ij}) can be expressed as follows

$$y_{(ij)}(k+1) = y_{(ij)}(k) + \rho(x_i(k) - x_j(k)) \tag{10}$$

for $\forall \quad e_{ij} \in \mathcal{E}$.

Lemma 2. *Under Assumption 2, $\forall i \in \mathcal{V}$, $f_i(x_i)$ can be rewritten as:*

$$\begin{aligned} f_i(x_i) = f_i(x_i(k)) + \nabla f_i(x_i(k))(x_i - x_i(k)) \\ + \frac{\Lambda_i(k)}{2}(x_i - x_i(k))^2, \end{aligned} \tag{11}$$

where $0 < \Lambda_i(k) \leq \sigma_i$. *And there exit a* $\theta \in (0,1)$ *make* $\Lambda_i(k) = \nabla^2 f_i(\xi)$ *with* $\xi = x_i(k) + \theta(x_i - x_i(k))$.

Proof. Under Assumption 2, $\forall i \in \mathcal{V}$, f_i is at least twice continuously differentiable. Then using the median theorem in the Taylor expansion of f_i, we can get the equation (11). Then, from Definitions 1 and 2 and Lemma 1, we can get $0 < \Lambda_i(k) \leq \sigma_i$. □

The Lemma 2 only ensures the existence and uniqueness of $\Lambda_i(k)$ for the function $f_i(x_i)$ at $x_i(k)$, and $\Lambda_i(k)$ is dependent on $\xi = x_i(k) + \theta(x_i - x_i(k))$ with $\theta \in (0,1)$. In the actual calculation, we are not able to determine the specific value of θ. So for the actual calculation of $\Lambda_i(k)$, what we can do is to approximate it with its estimation, and the specific treatment is as follows:

- When f_i is a second-order function, equation 11 is natural and $\Lambda_i(k)$ is just the second derivative $\nabla^2 f_i(x_i(k))$ which is just constant value and free of dependence on k and $x_i(k)$, then we can note $\Lambda_i(k) = \Lambda_i = \nabla f_i$.
- When f_i is a high-order function, due to the role of higher order residuals, $\Lambda_i(k) \neq \nabla^2 f_i(x_i(k))$, but the difference between the two is small. In the implementation, as $\nabla^2 f_i \in (0, \sigma_i]$, we can further relax the estimation and use $\Lambda_i(k) = \Lambda_i = \nabla^2 f_i(x_i(0))$ to approximate $\Lambda_i(k)$.

Thus, $\forall k$, each Λ_i is a positive constant that does not depend on k. We denote $\Lambda = \text{diag}\{\Lambda_1, \Lambda_2, \ldots, \Lambda_N\}$ and $\nabla F(\mathbf{x}) = (\nabla f_1(x_1), \nabla f_2(x_2), \ldots, \nabla f_N(x_N))^T$. Then the augmented Lagrangian $\mathcal{L}(\mathbf{x}, \mathbf{y}(k))$ can be calculated as follows:

$$\begin{aligned}
&\mathcal{L}_\rho(\mathbf{x}, \mathbf{y}) \\
&\simeq F(\mathbf{x}(k)) + \nabla F(\mathbf{x}(k))^T(\mathbf{x} - \mathbf{x}(k)) + \mathbf{y}(k)^T C\mathbf{x} \\
&\quad + \frac{1}{2}(\mathbf{x} - \mathbf{x}(k))^T \Lambda(\mathbf{x} - \mathbf{x}(k)) + \frac{\rho}{2}\mathbf{x}^T L\mathbf{x} \\
&= F(\mathbf{x}(k)) + \mathbf{y}(k)^T C\mathbf{x}(k) + \frac{\rho}{2}\mathbf{x}(k)^T \Lambda\mathbf{x}(k) \\
&\quad + (\nabla F(\mathbf{x}(k))^T + \mathbf{y}^T C + \rho\mathbf{x}(k)^T L)(\mathbf{x} - \mathbf{x}(k)) \\
&\quad + \frac{1}{2}(\mathbf{x} - \mathbf{x}(k))^T(\Lambda + \rho L)(\mathbf{x} - \mathbf{x}(k)).
\end{aligned} \tag{12}$$

Taking $\frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y}(k))}{\partial \mathbf{x}} = 0$, we get

$$\begin{aligned}
&(\Lambda + \rho L)(\mathbf{x} - \mathbf{x}(k)) \\
&= -(\nabla F(\mathbf{x}(k)) + C^T\mathbf{y}(k) + \rho L\mathbf{x}(k)).
\end{aligned}$$

Denote $\triangle\mathbf{x} = \mathbf{x} - \mathbf{x}(k)$ and

$$\mathbf{b}(k) = -(\nabla F(\mathbf{x}(k)) + C^T\mathbf{y}(k) + \rho L\mathbf{x}(k)). \tag{13}$$

The sub-problem (8) becomes the following linear equation problem:

$$H \triangle \mathbf{x} = \mathbf{b}(k), \tag{14}$$

where $H = \{h_{ij}\} = \Lambda + \rho L$.

The detailed calculation of i-th component $b_i(k)$ of $\mathbf{b}(k)$ can be expanded to:

$$\begin{aligned}
&b_i(k) \\
&= -\nabla f_i(x_i(k)) - \sum_{j \in \mathcal{N}_i} \rho(x_i(k) - x_j(k)) \\
&\quad - \left\{ \sum_{j=1}^{i} C_{((ji),i)} y_{(ji)}(k) + \sum_{j=i+1}^{N} C_{((ij),i)} y_{(ij)}(k) \right\} \\
&= -\nabla f_i(x_i(k)) - \sum_{j \in \mathcal{N}_i} \rho(x_i(k) - x_j(k)) \\
&\quad - \left\{ -\sum_{j=1}^{i} C_{((ji),i)} y_{(ij)}(k) + \sum_{j=i+1}^{N} C_{((ij),i)} y_{(ij)}(k) \right\}.
\end{aligned} \tag{15}$$

The second equal equation is due to $y_{(ij)}(k) = -y_{(ji)}(k)$, which comes from the mathematical induction based on equation (10) and the same initial value $y_{(ij)}(0) = -y_{(ji)}(0) = 0$. The detailed calculation of $b_i(k)$ depends only on the information of node i itself and its neighboring nodes, i.e.: $x_i(k)$ and $x_j(k)$, $y_{(ij)}(k)$ for $j \in \mathcal{N}_i$ and $\forall i \in \mathcal{V}$.

A key property about matrix H is as follows:

Lemma 3. *The matrix H above is strictly diagonally dominant, i.e.,* $h_{ii} > \sum_{j \neq i} |h_{ij}|$ *for* $\forall i \in \mathcal{V}$.

Proof. Denote $L = \{l_{ij}\}$. From $L = C^T C$ and the definition of C in (3), it is easy to verify that $l_{ij} < 0$ for all $i \neq j$ and that $l_{ii} = \sum_{j \neq i} |l_{ij}|$ for all i. Since Λ is a positive diagonal matrix, it follows that $h_{ii} = \Lambda_i + l_{ii} > \sum_{j \neq i} |l_{ij}| = \sum_{j \neq i} |h_{ij}|$ for $\forall i \in \mathcal{V}$. □

The strict diagonal dominance of the matrix H means that the matrix H is reversible. The following key problem is how to solve $\triangle\mathbf{x}$ from (14) in a distributed way.

3.2 | Generalized gradient algorithm

In the undirected connected simple graph $\mathcal{G}$, with $L = D - A$, we modify the equation (14) as

$$(\Lambda + \rho D) \triangle \mathbf{x} = \rho A \triangle \mathbf{x} + \mathbf{b}(k).$$

equivalently,

$$\triangle\mathbf{x} = (\Lambda + \rho D)^{-1}(\rho A \triangle \mathbf{x} + \mathbf{b}(k)). \tag{16}$$

The generalized gradient algorithm involves solving the above in an iterative form:

$$\triangle\mathbf{x}^{(t+1)} = (\Lambda + \rho D)^{-1}(\rho A \triangle \mathbf{x}^{(t)} + \mathbf{b}(k)) \tag{17}$$

with $t = 1, 2, \ldots$, and $\triangle\mathbf{x}^{(0)} = 0$. The distributed implementation of (17) is given by

$$\triangle x_i^{(t+1)} = (\Lambda_i + \rho d_{ii})^{-1} \left(\rho \sum_{j \in \mathcal{N}_i} \triangle x_j^{(t)} + b_i(k) \right). \tag{18}$$

We have the following key result for the generalized gradient algorithm.

Theorem 1. *Under Assumption 1 and 2, the spectral radius of matrix $(\Lambda+\rho D)^{-1}\rho A$ is strictly less than 1. Subsequently, the iterations in (17) (or (18)) convergences to (16) exponentially.*

Proof. From Lemma 3, $\Lambda + \rho L = \Lambda + \rho D - \rho A$ is strictly diagonally dominant. Since $\Lambda + \rho D$ is diagonal and positive, it follows that $(\Lambda + \rho D)^{-1}(\Lambda + \rho D - \rho A) = I - (\Lambda + \rho D)^{-1}\rho A$ is also strictly diagonally dominant. Note that the diagonal elements of A are zero. Denote $U = \{u_{ij}\} = (\Lambda + \rho D)^{-1}\rho A$. We have $\sum_{j\neq i}|u_{ij}| < 1$ for all i. By the well-known Gershgorin circle theorem [29], the eigenvalues of U are all inside the unit circle. Hence, the spectral radius of matrix $(\Lambda + \rho D)^{-1}\rho A$ is strictly less than 1. □

The result above shows that as the iteration $t \to \infty$, $\triangle\mathbf{x}^{(t)}$ in (17) converges to $\triangle\mathbf{x}$ in (16). This means that we can run (17) for m iterations to ensure a sufficiently accurate solution for $\triangle\mathbf{x}$ where d is the graph diameter and m is a constant much smaller than d. This leads to the generalized gradient-like iteration for solving the sub-problem (8):

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \triangle\mathbf{x}^{(m)}. \tag{19}$$

Together with (9), the generalized gradient based algorithm is summarized in Algorithm 1.

Algorithm 1 Distributed ADMM with Generalized Gradient

Initialization: Take any $\mathbf{x}(0)$ and $\rho > 0$ and set $\mathbf{y}(0) = 0$.

Main loop:

1: **for** $k = 1, 2, \cdots$ **do**
2: For each node i, compute $b_i(k)$ according to (15) and set $\triangle x_i^{(0)} = 0$.
3: **for** $t = 1, 2, \cdots, m$ **do**
4: For each node i, compute $\triangle x_i^{(t+1)}$ using (18).
5: **end for**
6: For each node i, update $x_i(k) = x_i(k-1) + \triangle x_i^{(m)}$.
7: For each edge (i, j) with $i < j$, update $y_{(i,j)}(k) = y_{(i,j)}(k-1) + \rho(x_i(k) - x_j(k))$.
8: **end for**

3.3 | Generalized Belief Propagation Algorithm

In this subsection, we introduce an alternative method, the generalized belief propagation (BP) algorithm, to solve the sub-problem (14). This algorithm is borrowed from [19].

For the given pair of $(H, \mathbf{b})$ and network $\mathcal{G}$, we define $h_{(i\to j)}(t)$ and $b_{(i\to j)}(t)$ to be two variables (or messages) to be passed from node i to node $j \in \mathcal{N}_i$ in the t-th inner iteration. Also defined are two internal variables $\tilde{h}_i(t)$ and $\tilde{b}_i(t)$ as well as the estimate $\triangle x_i^{(t)}$ for node i in the t-th inner iteration. These variables are initialized and iterated as follows: For every node $i \in \mathcal{V}$, node $j \in \mathcal{N}_i$, and iteration $t = 1, 2, \ldots$:

$$\tilde{h}_i(t) = h_{ii} - \sum_{v\in\mathcal{N}_i} \frac{h_{vi}h_{iv}}{h_{v\to i}(t-1)}, \tag{20}$$

$$\tilde{b}_i(t) = b_i(k) - \sum_{v\in\mathcal{N}_i} \frac{h_{iv}b_{v\to i}(t-1)}{h_{v\to i}(t-1)}, \tag{21}$$

$$\triangle x_i^{(t)} = \frac{\tilde{b}_i(t)}{\tilde{h}_i(t)}, \tag{22}$$

$$h_{i\to j}(t) = \tilde{h}_i(t) + \frac{h_{ji}h_{ij}}{h_{j\to i}(t-1)}, \tag{23}$$

$$b_{i\to j}(t) = \tilde{b}_i(t) + \frac{h_{ij}b_{j\to i}(t-1)}{h_{j\to i}(t-1)}, \tag{24}$$

with $h_{i\to j}(0) = h_{ii}$, $b_{i\to j}(0) = b_i(k)$.

We have the following key results for the generalized BP algorithm. The first one (Theorem 2) is a finite-time convergence result for acyclic graphs, and the second one is an asymptotic convergence result for general graphs. Their proofs are given in the next two subsections.

Theorem 2. *Under Assumption 1 and 2, with the additional condition that $\mathcal{G}$ is an acyclic graph with diameter d, it holds that*

$$\lim_{t\to d} \triangle x^{(t)} = H^{-1}\mathbf{b}(k) = \triangle x^*(k), \tag{25}$$

and

$$\triangle x^{(t)} = H^{-1}\mathbf{b}(k) = \triangle x^*(k), \tag{26}$$

for all $t \geq d$.

Theorem 3. *Under Assumption 1 and 2, it holds that*

$$\lim_{t\to\infty} \triangle\mathbf{x}^{(t)} = H^{-1}\mathbf{b}(k) = \triangle\mathbf{x}^*(k), \tag{27}$$

where $\triangle\mathbf{x}^{(t)} = col\quad\{x_1^{(t)}, x_2^{(t)}, \ldots, x_N^{(t)}\}$.

Similar to the generalized gradient algorithm, for a general connected graph, we can force the number of inner iterations m to be much smaller than the graph diameter d. This point will be illustrated in the next section.

Algorithm 2 Distributed ADMM with Generalized BP

Initialization: Take any $\mathbf{x}(0)$ and $\rho > 0$ and set $\mathbf{y}(0) = 0$.

Main loop:

1: **for** $k = 1, 2, \cdots$ **do**
2: For each node i: Compute $b_i(k)$ according to (15); Set $h_{i\to j}(0) = h_{ii}$, $b_{i\to j}(0) = b_i(k)$ and transmit them to each node $j \in \mathcal{N}_i$.
3: **for** $t = 1, 2, \cdots, m$ **do**
4: For each node i: Compute $\tilde{h}_i(t)$, $\tilde{b}_i(t)$ and $\triangle x_i^{(t)}$ according to (20)- (22);
5: Then for each $j \in \mathcal{N}_i$, compute $h_{i\to j}(t)$ and $b_{i\to j}(t)$ according to (23)- (24) and transmit them to node j.
6: **end for**
7: For each node i, update $x_i(k) = x_i(k-1) + \triangle x_i^{(m)}$.
8: For each edge (i, j) with $i < j$, update $y_{(i,j)}(k) = y_{(i,j)}(k-1) + \rho(x_i(k) - x_j(k))$.
9: **end for**

3.4 | Proof of Theorem 2

We first present a simple property for $h_{i\to j}(t)$ and $\tilde{h}_i(t)$ in Algorithm 2.

Lemma 4. *Suppose H is strictly diagonally dominant with positive digonals. Then, $h_{i\to j}(t) > |h_{ij}|$ and $\tilde{h}_i(t) > 0$ for all $i \in \mathcal{V}, j \in \mathcal{N}_i$ and $t = 0, 1, \ldots$. (Here we add the convention that $\tilde{h}_i(0) = h_{ii}$).*

Proof. We prove by induction. Since $h_{i\to j}(0) = \tilde{h}_i(0) = h_{ii}$, it is obvious that the result holds for $t = 0$. Suppose the result holds for some $t \geq 1$, i.e., $h_{i\to j}(t-1) > |h_{ij}|$ and $\tilde{h}_i(t-1) > 0$ for all $i \in \mathcal{V}, j \in \mathcal{N}_i$. We need to show that it also holds for t. Indeed,

$$\begin{aligned} h_{i\to j}(t) &= h_{ii} - \sum_{v\in\mathcal{N}_i\setminus j} \frac{h_{vi}h_{iv}}{h_{v\to i}(t-1)} \\ &\geq h_{ii} - \sum_{v\in\mathcal{N}_i\setminus j} |h_{iv}| \\ &= h_{ii} - \sum_{v\in\mathcal{N}_i} |h_{iv}| + |h_{ij}| > |h_{ij}|. \end{aligned}$$

Similarly,

$$\begin{aligned} \tilde{h}_i(t) &= h_{ii} - \sum_{v\in\mathcal{N}_i} \frac{h_{vi}h_{iv}}{h_{v\to i}(t-1)} \\ &\geq h_{ii} - \sum_{v\in\mathcal{N}_i} |h_{iv}| > 0. \end{aligned}$$

By induction, the claim holds for all $t = 0, 1, \ldots$. □

Now we are ready to prove Theorem 2.

Proof. We assume for now that $\mathcal{G}$ is connected and proceed to prove the theorem. Take any node and consider the tree representation of $\mathcal{G}$ with this node as the root. Because the ordering of the nodes does not affect the solution of (14), we assume, without loss of generality, that the root node is labeled as node 1. Thus, it suffices to show that $\triangle x_1^{(t)} = x_1^\star(k)$ for all $t \geq d$.

To help visualize the idea of the proof, we first discuss the simple graph in Figure 1 with $d = 2$. For this graph, we have

$$[H \mid \mathbf{b}] = \left[\begin{array}{ccccc|c} h_{11} & h_{12} & h_{13} & 0 & 0 & b_1 \\ h_{21} & h_{22} & 0 & h_{24} & h_{25} & b_2 \\ h_{31} & 0 & h_{33} & 0 & 0 & b_3 \\ 0 & h_{42} & 0 & h_{44} & 0 & b_4 \\ 0 & h_{52} & 0 & 0 & h_{55} & b_5 \end{array}\right] \tag{28}$$

Solving $\triangle x_1$ in $H \triangle \mathbf{x} = \mathbf{b}$ can be done by applying Gauss elimination step by step on $[H|\mathbf{b}]$ above.

Step 0.0 ($t = 0$): From the Initialization step in Algorithm 2, we can rewrite (28) as below in (29) to emphasize the transmissions from leaf nodes upwards (i.e., from node 3 to node 1 and from nodes 4 and 5 to node 2):

$$\left[\begin{array}{ccccc|c} h_{11} & h_{12} & h_{13} & 0 & 0 & b_1 \\ h_{21} & h_{22} & 0 & h_{24} & h_{25} & b_2 \\ h_{31} & 0 & h_{3\to1}(0) & 0 & 0 & b_{3\to1}(0) \\ 0 & h_{42} & 0 & h_{4\to2}(0) & 0 & b_{4\to2}(0) \\ 0 & h_{52} & 0 & 0 & h_{5\to2}(0) & b_{5\to2}(0) \end{array}\right] \tag{29}$$

Step 0.1 ($t = 1$): Adding a scaled version of row 3 to row 1 with scaling value of $-h_{13}/h_{33}$ will eliminate the (1,3)-entry of (28). Similarly, adding a scaled version of row 4 to row 2 with scaling value of h_{24}/h_{44} will eliminate the (2,4)-entry of (28), and adding a scaled version of row 5 to row 2 with scaling value of h_{25}/h_{55} will eliminate the (2,5)-entry of (28). The result becomes the following:

$$\left[\begin{array}{ccccc|c} h_{11} - \frac{h_{13}h_{31}}{h_{3\to1}(0)} & h_{12} & 0 & 0 & 0 & b_1 - \frac{h_{13}b_{3\to1}(0)}{h_{3\to1}(0)} \\ h_{21} & h_{22} - \frac{h_{42}h_{24}}{h_{4\to2}(0)} - \frac{h_{52}h_{25}}{h_{5\to2}(0)} & 0 & 0 & 0 & b_2 - \frac{h_{24}b_{4\to2}(0)}{h_{4\to2}(0)} - \frac{h_{25}b_{5\to2}(0)}{h_{5\to2}(0)} \\ h_{31} & 0 & h_{33} & 0 & 0 & b_3 \\ 0 & h_{42} & 0 & h_{44} & 0 & b_4 \\ 0 & h_{52} & 0 & 0 & h_{55} & b_5 \end{array}\right] \tag{30}$$

We see that the Gauss eliminations above effectively do the following: Add $-h_{13}h_{31}/h_{3\to1}(0)$ and $-h_{13}b_{3\to1}(0)/h_{3\to1}(0)$ to the (1,1)-entry and (1,6)-entry, respectively; And add $-(h_{24}h_{42}/h_{4\to2}(0) + h_{25}h_{52}/h_{5\to2}(0))$ and $-(h_{24}b_{4\to2}(0)/h_{4\to2}(0) + h_{25}b_{5\to2}(0)/h_{5\to2}(0))$ to the (2,2)-entry and (2,6)-entry, respectively. Note in (30) that nodes 1 and 2 are now uncoupled from nodes 3, 4 and 5. Also, because node 3 is a leaf node, we see from (23)- (24) that

$(h_{3\to1}(k), b_{3\to1}(k)) = (h_{3\to1}(0), b_{3\to1}(0))$ for all $k \geq 1$. Using (23)- (24), we can rewrite (30) as

$$\left[\begin{array}{ccccc|c} h_{11} & h_{12} & 0 & 0 & 0 & b_1 \\ \frac{h_{13}h_{31}}{a_{3\to1}(1)} & & & & & \frac{h_{13}b_{3\to1}(1)}{h_{3\to1}(1)} \\ \hline h_{21} & h_{2\to1}(1) & 0 & 0 & 0 & b_{2\to1}(1) \\ \hline h_{31} & 0 & h_{33} & 0 & 0 & b_3 \\ \hline 0 & h_{42} & 0 & h_{44} & 0 & b_4 \\ \hline 0 & h_{52} & 0 & 0 & h_{55} & b_5 \end{array}\right] \tag{31}$$

By Lemma 4, $h_{2\to1}(1) > |h_{21}|$.

Step 0.2 ($t = 2$): Adding a scaled version of row 2 to row 1 with scaling value of $-h_{12}/h_{2\to1}(1)$ will eliminate the (1,2)-entry of (31), giving the following equation for $\triangle x_1$:

$$\begin{aligned}&\left(h_1 - \frac{h_{12}h_{21}}{h_{2\to1}(1)} - \frac{h_{13}h_{31}}{h_{3\to1}(1)}\right) \triangle x_1 \\ =&b_1 - \frac{h_{12}b_{2\to1}(1)}{h_{2\to1}(1)} - \frac{h_{13}b_{3\to1}(1)}{h_{3\to1}(1)}.\end{aligned}$$

This is the same as transmitting $-h_{12}h_{21}/h_{2\to1}(1)$ and $-h_{12}b_{2\to1}(1)/h_{2\to1}(1)$ to node 1, and from (23)- (24), we get

$$\tilde{h}_1(2) \triangle x_1 = \tilde{b}_1(2).$$

By Lemma 4, $\tilde{h}_1(2) > 0$. This confirms that $\triangle x_1^{(2)}$ in (22) coincides with $\triangle x_1^\star(k)$. Similar to Step 0.2, we see that $(h_{2\to1}(t), b_{2\to1}(t)) = (h_{2\to1}(1), b_{2\to1}(1))$ for all $t \geq 2$. Therefore, $\triangle x_1^{(t)} = \triangle x_1^{(2)}$ for all $t > 2$.

We now consider a general connected acyclic graph $\mathcal{G}$ with diameter d. Denote by τ the depth of the tree graph, i.e., the longest path from a leaf node to the root node (node 1). it is clear that $\tau \leq d$. Without loss of generality, assume that the nodes are listed in the following order: All the leaf nodes are at the bottom of the list; Once all the leaf nodes are removed, the leaf nodes of the remaining graph are at the bottom of the remaining list; and so on. As before, we study the steps of Gauss elimination for computing $\triangle x_1^\star$.

Step 0 ($t = 0$): Let node i be any leaf node and node j be its (unique) neighboring node. Then, by the Initialization step in Algorithm 2, we have $h_{i\to j}(0) = h_{ii}$ and $b_{i\to j}(0) = b_i$.

Step 1 ($t = 1$): Again, let node i be any leaf node and node j be its (unique) neighboring node. Adding a scaled version of row i to row j with the scaling parameter of $-h_{ji}/h_{i\to j}(0)$ will eliminate the (j,i)-entry of the matrix $[H|\mathbf{b}]$. As verified in the example in Figure 1, this operation is equivalent to transmitting $-h_{ji}h_{ij}/h_{i\to j}(0)$ and $-h_{ji}b_{i\to j}(0)/h_{i\to j}(0)$ to the (j,j)-entry and $(j,(n+1))$-entry, respectively. After the Gauss elimination, all the leaf nodes are decoupled from the non-leaf nodes in $\mathcal{G}$. Similar to Step 0.1 above, we have $a_{i\to j}(t) = a_{i\to j}(0)$ and $b_{i\to j}(t) = b_{i\to j}(0)$ for all $t > 0$, due to the fact that i is a leaf node.

Step 2 ($t = 2$): Consider the reduced graph with all the leafs removed and the remaining matrix of the modified $[H|\mathbf{b}]$. Denote by n_1 the remaining number of nodes. Let node i be any new leaf (which is the neighbor of an old leaf) and let j be the (unique) neighbor of i. From Step 1 and (20)- (21), we see that the (i,i)-entry is actually equal to $h_{i\to j}(k-1)$. Similarly, the $(i,(n_1+1))$-entry equals to $b_{i\to j}(k-1)$. Also similar to the example above, we note from Lemma 4 that $h_{i\to j}(k-1) > |h_{ij}|$ and $\tilde{h}_i(k-1) > 0$. We can apply the Gauss elimination in Step 1 again. .....

Step t ($t \geq \tau$): The above process can be repeated until only the root node remains. Similar to the graph in Figure 1, it is tedious but straightforward to verify that the resulting $\triangle x_1^\star$ is indeed given by $\triangle x_1^{(\tau)}$ in (22). Also note from Lemma 4 that $\tilde{h}_1(\tau) > 0$. It is similar to Steps 0.1-0.2 that $\triangle x_1^{(t)} = \triangle x_1^{(\tau)}$ and $\tilde{h}_1(t) = \tilde{h}_1(\tau)$ for all $t > \tau$. Since $\tau \leq d$, the above means that $\triangle x_1^{(t)} = \triangle x_1^{(d)}$ and $\tilde{h}_1(t) = \tilde{h}_1(d)$ for all $t > d$.

Since the root node is arbitrarily chosen, we conclude that $\triangle \mathbf{x}^{(t)} = \triangle \mathbf{x}^{(d)} = \triangle x^\star$ for all $t > d$.

Finally, we consider the general case of an acyclic $\mathcal{G}$ which is not necessarily connected. If $\mathcal{G}$ is disconnected, then $\mathcal{G}$ can be decomposed of a finite number of disjoint subgraphs $\mathcal{G}^i$, each being connected with its diameter $d^i \leq d$. By definition, d is also the diameter of one of the subgraphs. Accordingly, matrix H can be transformed through row and column permutations such that the resulting matrix H is decomposed of block diagonal matrices H^i, i.e., (8) can be rewritten as a finite set of $H^i \triangle \mathbf{x}^i = \mathbf{b}^i$. We see that running Algorithm 2 is effectively running the algorithm on each $\mathcal{G}^i$. Applying the proved results above on a connected graph, we conclude again that $\triangle \mathbf{x}^{(t)} = \triangle \mathbf{x}^{(d)} = \triangle x^\star$ for all $t > d$. □

3.5 | Proof of theorem 3

Algorithm 2 is borrowed from the Gaussian BP algorithm in [21] (For details, see [21]). The following convergence property is given in [21].

Lemma 5. *Suppose H consists of pairwise cliques (also called pairwise normalizable form [30]), i,e.,*

$$H = \sum_{e=(i,j)\in\mathcal{E}} [H_{(e)}], \tag{32}$$

where $\mathcal{E}$ is the edge set corresponding to the matrix H, $H_{(e)}$ is a symmetric and positive-definite 2×2 matrix, $[H_{(e)}]$ is a zero-padded matrix of the size H such that the $(i,i),(i,j)(j,i)$ and (j,j)-th elements correspond to those of

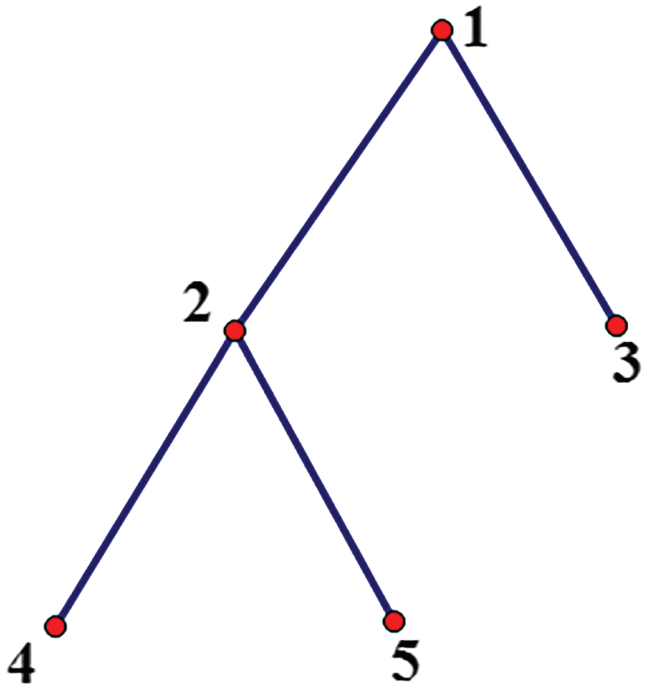

FIGURE 1 Illustration for the proof of Theorem 2 [Color figure can be viewed at wileyonlinelibrary.com]

$H_{(e)}$. Then, the asymptotic convergence property (27) for Algorithm 2 holds.

It turns out that the pairwise clique form is guaranteed under the strictly diagonal dominance condition, as shown below.

Lemma 6. *If a matrix H has positive diagonal elements and is strictly diagonally dominant, then it can be written in the form of (32).*

Proof. Using the strictly diagonal dominance property, we can express, for each row i of H,

$$h_{ii} = \sum_{j \in \mathcal{N}_i} (|h_{ij}| + \delta_{ij})$$

for some (sufficiently small) $\delta_{ij} > 0$ for all j above. With this, it is straightforward to see that H can be rewritten as (32) with

$$H_{(e)} = \begin{bmatrix} |h_{ij}| + \delta_{ij} & h_{ij} \\ h_{ji} & |h_{ji}| + \delta_{ji} \end{bmatrix}$$

for each $e = (i, j) \in \mathcal{E}$. It is clear that $H_{(e)}$ has positive diagonals and determinant. □

Powered by the above results, we can proceed to prove Theorem 3.

Proof. The result is a direct combination of Lemmas 5-6. □

3.6 | Algorithm analysis and summary

Our Algorithms 1 and 2 are respectively embedded with generalized gradient algorithm and generalized Belief Propagation algorithm under the ADMM framework [13]. Our cost functions are closed, proper, and convex, and the unaugmented Lagrangian L_ρ has a saddle point (which is guaranteed by the existence and uniqueness of the solution of the original problem), then these two proposed algorithms can converge by proper parameter(ρ) adjustment.

Different from the existing gradient descent method and more complex dual method, our method builds linear equations based on optimality conditions, and realizes the iteration of our algorithm by using the distributed fast solving algorithm for linear equations. Although our BP-based algorithm requires two additional auxiliary variables compared with the generalized gradient method using jacobian iteration, it requires fewer internal iterations at the same time. Subsequent simulations show that our algorithm requires only a small number of internal iterations to ensure linear convergence.This shows that compared with the standard single-layer gradient descent method, our two algorithms do not add much complexity in practical application.

4 | SIMULATION AND PERFORMANCE ANALYSIS

In this section, we study the performances of Algorithms 1-2.

4.1 | Quadratic objective functions case

The objective function is given by $f(x) = \frac{1}{2}\sum_{i=1}^{N} q_i x^2 + p_i x$ with $\nabla^2 f(x) = q_i > 0$. The linear terms $p_i x$ are added so that the different local functions have different minima. The values of q_i and p_i are randomly from $[1, 50]$ and $[-1, 1]$, respectively. The graph we consider a random connect network with a group of 200 agents and 400 edges. In order to analyze the convergence rate of each algorithm, we consider the MSE (mean-square error) which is defined as

$$\text{MSE} \quad (k) = \frac{1}{N}\sum_{i=1}^{N} \frac{\|x_i(k) - x^*\|}{\|x_i(0) - x^*\|}. \tag{33}$$

Figures 2-4 show the simulation results for $m = 1, 2, 6$ with $\rho = 10$, respectively, and where m represents the number of inner iterations. In each figure, MSE-G and MSE-BP correspond to the generalized gradient algorithm and the generalized BP algorithm, respectively. And the speed $1/k^2$ is achieved by an accelerated distributed Nesterov gradient method with multi-steps consensus inner iterations in [6] for convex and smooth cost functions. A number of observations are in order.

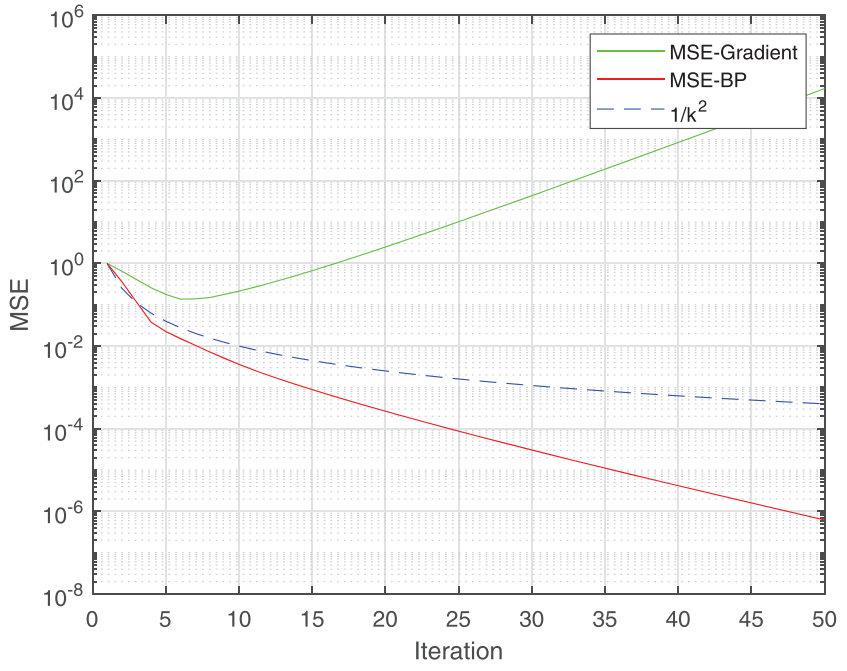

FIGURE 2 MSEs with $m = 1$ [Color figure can be viewed at wileyonlinelibrary.com]

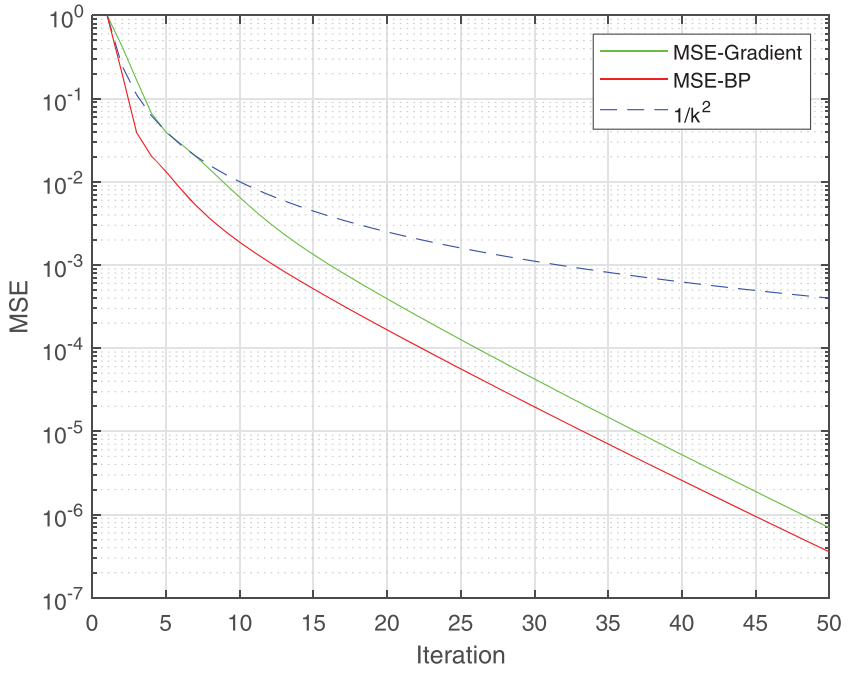

FIGURE 3 MSEs with $m = 2$ [Color figure can be viewed at wileyonlinelibrary.com]

- Firstly, $m = 1$, the generalized gradient algorithm is divergent and the generalized BP algorithm is convergent under the same parameters.
- Secondly, as $m \geq 2$ (the inner iteration number) increases, both algorithms converge faster than $1/k^2$. But when m becomes large, the incremental benefit diminishes. For example, the difference between $m = 2$ and $m = 6$ for the generalized BP algorithm is marginal if MSE of 10^{-4} is required.
- Thirdly, the generalized BP algorithm significantly outperforms that of generalized gradient algorithm, especially when m is small. We see that the performance of the generalized BP algorithm for $m = 2$ is on par with that of the generalized gradient algorithm for $m = 6$.
- Finally, with a small number of inner iterations ($m \geq 2$), both algorithms can be regarded to have linear convergence.

From the above analysis, under fewer inner iterations, we see that the generalized BP algorithm clearly outperforms the generalized gradient algorithm. In particular, our BP-based algorithm can achieve linear convergence in

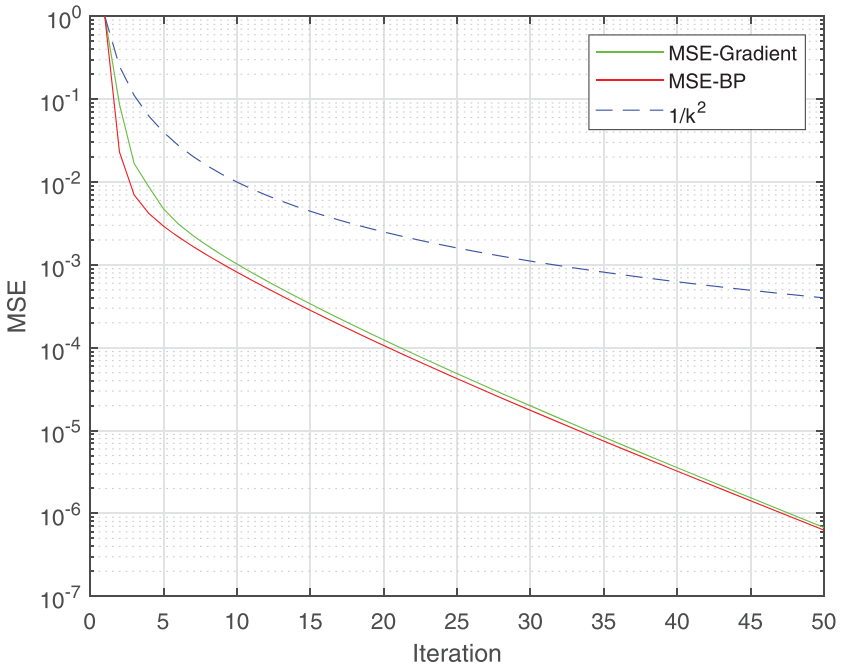

FIGURE 4 MSEs with $m = 6$ [Color figure can be viewed at wileyonlinelibrary.com]

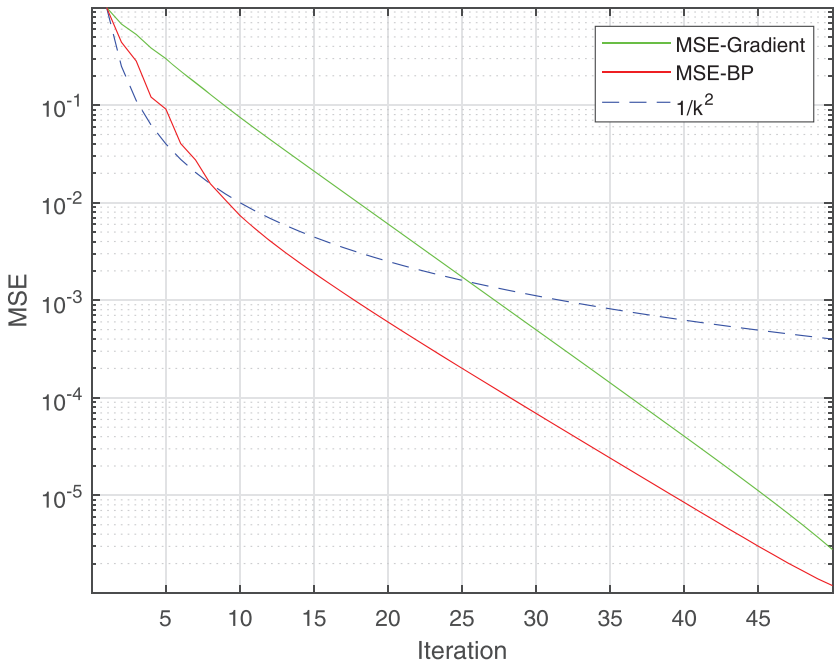

FIGURE 5 MSEs with $m = 2$ [Color figure can be viewed at wileyonlinelibrary.com]

only one step of inner iteration. In addition, in both algorithms, there is a clear tradeoff between the inner and outer loops. One or two steps of internal iteration is the most appropriate, more internal iteration will only increase the complexity of computation and information exchange, but will not greatly improve the convergence effect

4.2 | Higher order nonlinear objective function case

In the same fixed graph, if the function is not a quadratic form, but with the following form: each node i observes a function $f_i : R \to R$, given by

$$f_i(x) = \hat{a}_i x + \hat{b}_i(x - \hat{c}_i)^2 + \hat{d}_i(x - \hat{e}_i)^4,$$

where $\hat{a}_i, \hat{b}_i, \hat{c}_i, \hat{d}_i, \hat{e}_i$ are parameters of f_i,whose values are randomly chosen from the intervals $(-1, 1), (0, 1)(-1, 1), (0, 2), (-1, 1)$. As $\hat{b}_i$ and $\hat{d}_i$ are positive, we can easily confirm that the function f_i satisfies Assumption 2. In the main equation (14), the Λ_i in $H = \Lambda + \rho(D - A)$ is selected as an approximate estimate: $2\hat{b}_i + 12\hat{d}_i * \hat{e}_i^2$ where $\nabla^2 f_i(x_i) = 2\hat{b}_i x_i + 12\hat{d}_i(x_i - \hat{e}_i)^2$. The iteration initial value is randomly selected in the range $(-2, 2)$. Figure 5 shows the simulation results for MSE-G

and MSE-BP with inner iteration $m = 2$ and $\rho = 10$. Two observations are following:

- Although the simulation effect of both algorithms are slightly worse than the corresponding quadratic case under the same parameters, the convergence speed of both algorithms will eventually exceed the curve of $1/k^2$.
- In the comparison of the two algorithms, the generalized BP algorithm has a better convergence performance than the generalized gradient algorithm. And both algorithms give linear convergence.

In summary, in the case of very small internal iterations $m = 2$ (without adding too much information exchange, calculation and storage relative to the single-layer gradient method), our two methods can be regarded to have linear convergence on dealing with large-scale nonlinear optimization problems. Our methods require less information exchange, calculation and storage relative to the existing (accelerated) gradient descent method (with convergence rate $1/k^2$) with the same moderate precision requirements(e.g., 10^{-3}).

5 | CONCLUSION

We have presented two ADMM-based distributed algorithms for convex optimization over large networks, one using the commonly used gradient approach, and one using the Gaussian belief propagation method. Theoretical properties and comparative studies are done for both algorithms. Emphasis is on the generalized BP algorithm which, although having more complex presentation, has better performance in comparison with the gradient approach.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (61633014, 61573221 and U1701264)

REFERENCES

1. T. Han et al., *Distributed three dimensional formation containment control of multiple unmanned aerial vehicle systems*, Asian J. Control **19** (2017), no. 3, 1103–1113.
2. B. Johansson. (2008). *On distributed optimization in networked systems*, Ph.D. Thesis, KTH, Stockholm.
3. S. Zhu et al., *Distributed optimal consensus filter for target tracking in heterogeneous sensor networks*, IEEE Trans. Cybern. **43** (2013), no. 6, 1963–1976.
4. J. B. Predd, S. R. Kulkarni, and H. V. Poor, *A collaborative training algorithm for distributed learning*, IEEE Trans. Inform. Theory **55** (2009), no. 4, 1856–1871.
5. A. Nedic and A. Ozdaglar, *Distributed subgradient methods for multi-agent optimization*, IEEE Trans. Autom. Control **54** (2009), no. 1, 48–61.
6. D. Jakovetić, J. Xavier, and J. M. F. Moura, *Fast distributed gradient methods*, IEEE Trans. Autom. Control **59** (2014), no. 5, 1131–1146.
7. G. Qu and N. Li, *Accelerated distributed nesterov gradient descent for convex and smooth functions*, 2017 IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, VIC, Australia, 2017, pp. 2260–2267.
8. G. Qu and N. Li, *Harnessing smoothness to accelerate distributed optimization*, IEEE Trans. Contr. Netw. Syst. **5** (2018), no. 3, 1245–1260.
9. R. Xin and U. A. Khan, *Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking*, 2018. arXiv preprint arXiv:1808.02942.
10. W. Shi et al., *Extra: An exact first-order algorithm for decentralized consensus optimization*, SIAM J. Optim. **25** (2015), no. 2, 944–966.
11. D. Jakovetić, *A unification and generalization of exact distributed first-order methods*, IEEE Trans. Signal Inform. Proces. over Netw. **5** (2018), no. 1, 31–46.
12. A. Mokhtari, Q. Ling, and A. Ribeiro, *Network newton distributed optimization methods*, IEEE Trans. Signal Proces. **65** (2017), no. 1, 146–161.
13. S. Boyd et al., *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends® Machine Learn. **3** (2011), no. 1, 1–122.
14. T. Chang, M. Hong, and X. Wang, *Multi-agent distributed optimization via inexact consensus ADMM*, IEEE Trans. Sign. Proces. **63** (2015), no. 2, 482–497.
15. D. Jakovetić, J. M. F. Moura, and J. Xavier, *Linear convergence rate of a class of distributed augmented lagrangian algorithms*, IEEE Trans. Autom. Control **60** (2015), no. 4, 922–936.
16. T. Chang, *A proximal dual consensus ADMM method for multi-agent constrained optimization*, IEEE Trans. Sign. Proces. **64** (2016), no. 14, 3719–3734.
17. P. Giselsson and S. Boyd, *Linear convergence and metric selection for Douglas-Rachford splitting and ADMM*, IEEE Trans. Autom. Contr. **62** (2017), no. 2, 532–544.
18. S. Wei et al., *On the linear convergence of the ADMM in decentralized consensus optimization*, IEEE Trans. Signal Proces. **62** (2014), no. 7, 1750–1761.
19. Q. Cai, Z. Zhang, and M. Fu, *A fast converging distributed solver for linear systems with generalised diagonal dominance*, 2019. arXiv:1904.12313.
20. J. Pearl, *Probabilistic reasoning in intelligent systems*, Morgan Kaufman, San Francisco, CA, USA, 1988.
21. Y. Weiss and W. T. Freeman, *Correctness of belief propagation in gaussian graphical models of arbitrary topology*, Neural Comput. **13** (2001), no. 10, 2173–2200.
22. W. Ma et al., *Finite-time average consensus based approach for distributed convex optimization*, Asian J. Control **21** (2019Nov), no. 6, 1–11.
23. B. Ning et al., *Distributed fixed-time coordinated tracking for nonlinear multi-agent systems under directed graphs*, Asian J. Control **20** (2018), no. 2, 646–658.
24. F. Wang et al., *Finite-time consensus problem for second-order multi-agent systems under switching topologies*, Asian J. Control **19** (2017), no. 5, 1756–1766.

25. Q. Wang, Y. Wang, and C. Sun, *Fixed-time consensus of multi-agent systems with directed and intermittent communications*, Asian J. Control **19** (2017), no. 1, 95–105.
26. S. S. Alaviani and N. Elia, *A distributed algorithm for solving linear algebraic equations over random networks*, 2018. arXiv:1809.07955.
27. J. Liu et al., *Exponential convergence of a distributed algorithm for solving linear algebraic equations*, Automatica **83** (2017), 37–46.
28. S. Mou, J. Liu, and A. S. Morse, *A distributed algorithm for solving a linear algebraic equation*, IEEE Trans. Autom. Control **60** (2015), no. 11, 2863–2878.
29. G. H. Golub and C. F. V. Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, 1996.
30. D. M. Malioutov, J. K. Johnson, and A. S. Willsky, *Walk-sums and belief propagation in gaussian graphical models*, J. Machine Learn. Res. **7** (2006), 2031–2064.

AUTHOR BIOGRAPHIES

Wenlong Ma received his B.S. degree and M.S. degree in Mathematics, from Shangqiu Normal University in 2012 and Shandong University, Jinan, China, in 2015, respectively. Since 2015 he is pursuing his Ph.D. degree at the School of Control Science and Engineering, Shandong University. His research interests include multi-agent network consensus control and distributed optimization.

Huanshui Zhang received the B.S. degree in mathematics from Qufu Normal University, Shandong, China, in 1986, the M.Sc. degree in control theory from Heilongjiang University, Harbin, China, in 1991, and the Ph.D. degree in control theory from Northeastern University, China, in 1997. He was a Postdoctoral Fellow at Nanyang Technological University, Singapore, from 1998 to 2001 and Research Fellow at Hong Kong Polytechnic University, Hong Kong, China, from 2001 to 2003. He is currently holds a Professorship at Shandong University, Shandong, China. He was a Professor with the Harbin Institute of Technology, Harbin, China, from 2003 to 2006. He also held visiting appointments as a Research Scientist and Fellow with Nanyang Technological University, Curtin University of Technology, and Hong Kong City University from 2003 to 2006. His interests include optimal estimation and control, time-delay systems, stochastic systems, signal processing and wireless sensor networked systems.

Minyue Fu received his Bachelor's Degree in Electrical Engineering from the University of Science and Technology of China, Hefei, China, in 1982, and M.S. and Ph.D. degrees in Electrical Engineering from the University of Wisconsin-Madison in 1983 and 1987, respectively. He joined the Department of Electrical and Computer Engineering, the University of Newcastle, Australia, in 1989. Currently, he is a Chair Professor in Electrical Engineering and Head of School of Electrical Engineering and Computer Science. In addition, he was a Visiting Associate Professor at University of Iowa in 1995-1996, and a Senior Fellow/Visiting Professor at Nanyang Technological University, Singapore, 2002. He has held a Qian-ren Professorship at Zhejiang University and Guangdong University of Technology, China. He is a Fellow of IEEE. His main research interests include control systems, signal processing and communications. He has been an Associate Editor for the IEEE Transactions on Automatic Control, Automatica and Journal of Optimization and Engineering.

How to cite this article: Ma W, Zhang H, Fu M. Distributed convex optimization based on ADMM and belief propagation methods. *Asian J Control*. 2020;1–12. https://doi.org/10.1002/asjc.2284