Asian Journal of Control, Vol. 21, No. 6, pp. 1–11, November 2019
Published online in Wiley Online Library (wileyonlinelibrary.com) DOI: 10.1002/asjc.1886

FINITE-TIME AVERAGE CONSENSUS BASED APPROACH FOR DISTRIBUTED CONVEX OPTIMIZATION

Wenlong Ma, Minyue Fu, Peng Cui, Huanshui Zhang and Zhipeng Li

ABSTRACT

In this paper, we consider a distributed convex optimization problem where the objective function is an average combination of individual objective function in multi-agent systems. We propose a novel Newton Consensus method as a distributed algorithm to address the problem. This method utilises the efficient finite-time average consensus method as an information fusion tool to construct the exact Newtonian global gradient direction. Under suitable assumptions, this strategy can be regarded as a distributed implementation of the classical standard Newton method and eventually has a quadratic convergence rate. The numerical simulation and comparison experiment show the superiority of the algorithm in convergence speed and performance.

Key Words: Newton method, finite-time average consensus, distributed convex optimization, multi-agent systems, quadratic convergence.

I. INTRODUCTION

In recent years, distributed convex optimization has been of considerable interest in distributed control and coordination of networks consisting of multiple autonomous agents (nodes). In this paper, we study distributed algorithms for solving convex optimization problems over networks. Our main concern is to determine a global decision variable (vector) $x \in R^n$ that minimizes an objective function $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$ in a multi-agent network. Each $f_i(x)$ represents a local objective function at node i. This problem has found various applications in multi-agent control [1], sensor fusion in wireless sensor network [2], and distributed learning [3] to just name a few. A notable feature of these problems is that the number of component functions is typically large and each component function is available only to an individual agent. Therefore, it is of great application value and theoretical significance to seek for a distributed algorithm with fast convergence for solving the convex optimization problem.

In the existing literature, gradient based methods are widely used for solving the above convex optimization problem. These methods are based on classic works on convex optimization [4] and [5]. The poineering work on the gradient based distributed optimization approach is the distributed (sub-)gradient method (DSM) in [6]. If $f(x)$ is convex, continuously differentiable, and its gradient is Lipschitz continuous with constant L, the sequence $x(k)$ generated by gradient descent method converges at rate is $O(1/k)$ with k being the iteration number. In the early 1980s, Nemirovski and Yudin [7] proved that no first-order method can converge at a rate faster than $O(1/k^2)$ for convex optimization problems with Lipschitz continuous gradient. This critical convergence rate was proved to be achievable by Nesterov, who presented a first-order gradient descent method that converges as $O(1/k^2)$ in [8]. With the Lipschitz continuous gradient and strongly convex objective function conditions, the Nesterov gradient method converges linearly in [9]. The accelerated heavy-ball algorithm in [10], and accelerated distributed Nesterov gradient method in [11] can achieve a better (smaller) convergence factor than the one associated with the gradient iterates, but the convergence rate is still within the framework of linear convergence. Thus, if we want to further improve the convergence rate and obtain a super-linear convergence rate, we need to consider using the second-order gradient information of the objective function.

Many second-order gradient methods are also available for distributed optimization. In [12], an inexact distributed Newton-type algorithm with the matrix splitting technique and dual frame to solve network utility

Manuscript received November 23, 2017; revised April 16, 2018; accepted June 3, 2018.

Wenlong Ma, Peng Cui, Zhipeng Li(lizhipeng116@163.com), and Huanshui Zhang are with the School of Control Science and Engineering, Shandong University, Jinan, 250061, China.

Minyue Fu (corresponding author, e-mail: minyue.fu@newcastle.edu.au) is with the School of Electrical Engineering and Computing, University of Newcastle, NSW 2308 Australia. He also holds a Qianren Professorship at the School of Automation, Guangdong Unversity of Technology, Guangzhou, China.

This work is supported by the Taishan Scholar Construction Engineering by Shandong Government, the National Natural Science Foundation of China (61633014, 61573221).

maximization (NUM) problems was developed. In [13], the Newton distributed algorithm is based on the Taylor expansion of the inverse of the Hessian matrix. In [14], an approximate global Newton-Raphson gradient direction is gradually constructed by means of average consensus protocols. The convergence of these Newton-like methods is at least superlinear under certain assumptions.A common feature of the aforementioned literature on the second-order gradient methods is the use of inaccurate (approximate) Newton directions. This direction is very convenient for distributed computing through matrix decomposition or consensus methods However, the inaccurate iterative direction also causes the convergence rate to be significantly slower than using accurate Newton directions, thus requiring a large number of iterative steps to reach the same accuracy, causing a great burden of calculation and information exchange. In this sense, the distributed implementation of the accurate Newton method can improve the convergence speed and reduce complexity.

In this paper, based on the second-order gradient method, we develop a distributed exact Newton algorithm to solve the above convex optimization problem. In the process of distributed implementation, average consensus theory is a particularly suitable tool as it allows agents to obtain global information by taking local actions over networks [15] and [16]. Due to the agreement based on the classic Laplacian matrix, the consensus reached is only asymptotic. Recently, there have been many attempts to achieve finite-time consensus, for example, a sequence of time-varying stochastic matrices were considered in [17], finite-time average consensus via iterated max-consensus in [18], a distributed continuous-time protocol in paper [19] and a new class of nonlinear protocols in [20,21]. In our proposed algorithm, we speed up the average consensus process by replacing the Laplacian matrix based method with a new and powerful model for information exchange and calculation in [22] to reach the exact average consensus in a finite number of iterations.

The details of the proposed algorithm are as follows. Based on the work in article [14], the algorithm is also divided into two layers of iterations. The outer layer is a standard Newton iteration. In each Newton iteration, we insert the inner layer where we use a finite-time consensus algorithm to construct an accurate Newtonian-Raphson gradient direction. After a finite number of iterations of the outer layer, the algorithm will reach a quadratic convergence rate. The proposed algorithm is a distributed implementation of an exact Newton method, which is the biggest difference from the approximate Newton method in [14].

The rest of the paper is organized as follows. In Section II, we define notations, the relevant algebraic graph theory, and the relevant lemma invoked in the paper, and formally state our problem and necessary assumptions in Section III. In Section IV, we show our main results on the distributed optimization algorithm. In Section V, we simulate the performance of our algorithm and compare it with other distributed algorithms. We conclude the paper in Section VII.

II. PRELIMINARIES

2.1 Basic notations

Let R and N_+, respectively, be the set of real and natural numbers. For a matrix $A \in R^{m\times m}$, we denote its transpose matrix by A^T. Let 1_n denote the vector of n ones, and $\mathbf{I}_n$ (or simply $\mathbf{I}$) denote the $n \times n$ identity matrix. For matrices A and B, the Kronecker product is denoted by $A \otimes B$. For symmetric matrices A and B, $A \succ B$ ($A \succeq B$) indicates that matrix($A - B$) is positive definite (semi-definite). The standard Euclidean norm of vector $x \in R^n$ is denoted by $\|x\|$. we use subscripts, for example, $f_i(x_i)$ to denote the local function of node i with its own local state x_i, the aggregated form of x_i is the vector $\mathbf{x} = [x_1^T, x_2^T, \cdots, x_N^T]^T$.

2.2 Graph theory

We briefly review some basic concepts from algebraic graph theory following [15]. A graph (a network) can be expressed as a triplet $G = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V} = \{1, \cdots, N\}$ is the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and A is the adjacency matrix which is defined as $a_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and $a_{ij} = 0$, otherwise. A graph is undirected if $(i,j) \in \mathcal{E}$ anytime $(j,i) \in \mathcal{E}$. An undirected graph is called connected if there is path from every node in $\mathcal{V}$ to every other node.

2.3 Classical optimization theory

Lemma 1. Suppose f icfferentiable, $\nabla^2 f(x)$ is positive definite and upper bounded. Let x^* is the global optimal decision, For $\delta > 0$, let S_δ denote the sphere $\{x \mid \|x - x^*\| \leq \delta\}$. Then

1. Consider a sequence $\{x(k)\}$ generated by the Newton gradient iteration:

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

if exists $\delta > 0$ and $x(0) \in S_\delta$, Then, $x(k) \in S_\delta$ and with the super-linear convergence rate

$$\|x(k+1) - x^*\| \le C\|x(k) - x^*\|^p$$

with $p > 1$, where C is a constant.

2. If for some $L > 0$, $Q > 0$, $\delta > 0$, and for all x and y in S_δ,

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \le L\|x - y\|$$

$$\|(\nabla^2 f(x))^{-1}\| \le Q$$

then, if $x(0) \in S_\delta$, we have

$$\|x(k+1) - x^*\| \le \frac{LQ}{2}\|x(k) - x^*\|^2$$

so if $\frac{LQ\delta}{2} < 1$ and $x(0) \in S_\delta$, $\|x(k) - x^*\|$ converges super-linearly with order at least two, that is, the Newton gradient iteration is quadratic convergence.

This result is standard in [5] (e.g., Proposition 1.4.1).

III. PROBLEM FORMULATION

Now we consider a scenario where nodes cooperatively minimize a global system objective. In a network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}\}$ of N nodes, each node is associated with a cost function $f_i : R^n \mapsto R$ which is a convex objective function known only by node i, and $f : R^n \mapsto R$ is a well-defined global cost function. Our overarching goal is to solve the optimization problem

$$x^* = arg \min_{x \in R^n} f(x) = arg \min_{x \in R^n} \frac{1}{N} \sum_{i=1}^{N} f_i(x). \tag{1}$$

Assumption 1. The undirected and connected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ has a support tree and no loops.

Remark 1. For a loopy graph, we just need to increase an algorithm process of finding the support tree in order to apply our results, the standard algorithms are depth-first search and breath-first search in graph theory [23] with complexity $O(|\mathcal{V}| + |\mathcal{E}|)$. And [22] proposed a new depth-first search algorithm, which can handle these operations in a distributed way with low complexity.

Assumption 2. All the local function $f_i(x)$ are twice differentiable and have bounded and strictly positive definite Hessian, which means that the eigenvalues of the local Hessians are within the interval $[m, M]$ where $0 < m \le M \le \infty$, that is, $0 \prec m\mathbf{I} \preceq \nabla^2 f_i(x) \preceq M\mathbf{I}, \forall x \in R^n$.

Remark 2. If f is twice continuously differentiable and there exist constants $m > 0$ such that $\nabla^2 f(x) \succeq m\mathbf{I}$ for all $x \in R^n$ which means f is strongly convex, then the optimal decision value x^* is unique.

Assumption 3. The local objective function Hessians $\nabla^2 f_i(x)$ are Lipschitz continuous with parameter L. I.e., $\|\nabla^2 f_i(x) - \nabla^2 f_i(y)\| \le L\|x - y\|, \forall x, y \in R^n$.

From the above two assumptions, we can naturally introduce the following proposition.

Proposition 1. Under Assumption 2 and 3, as every cost function $f_i(x)$ holds a bounded positive definite Hessian and $\nabla^2 f_i(x)$ is Lipschitz continuous with L, then function $f(x)$ also satisfies $0 \le m\mathbf{I} \le \nabla^2 f(x) \le M\mathbf{I}$, $\|\nabla^2 f(x) - \nabla^2 f(y)\| \le L\|x - y\|$ and $\forall x, y \in R$, and $\|(\nabla^2 f(x))^{-1}\|$ is bounded with $Q(= \frac{1}{m})$.

IV. MAIN RESULTS

The main problem (1) is an unconstrained optimization problem, which holds a global decision variable in the whole network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. As the network is connected, the main problem can be rewritten as an equivalent form:

$$\begin{aligned} x^* &= \arg\min_{x \in R^n} \frac{1}{N} \sum_{i=1}^{N} f_i(x_i) \\ s.t. \quad & x_i = x_j, \forall i \in \mathcal{V}, j \in \mathcal{N}_i. \end{aligned} \tag{2}$$

where the constraints is equivalent to $x_i = \bar{x} = \frac{\sum_i x_i}{N}$ for all $i \in \mathcal{V}$.

4.1 Newton optimization

In the following, we mainly consider this equivalence problem (2), first, putting away the constrains temporarily, then we get a series of optimal value points $\{x_i^*\}_{i=1}^N$ where

$$\{x_i^*\}_{i=1}^N \doteq \arg\min_{x \in R^n} \frac{1}{N} \sum_{i=1}^{N} f_i(x_i), \forall i \in \mathcal{V}. \tag{3}$$

As the objective function is separable now, based on pure standard Newton iteration (where the step size in every iteration is unit), every single node $i \in \mathcal{V}$ (decision-maker) could simply minimize its own cost independently, that is,

$$x_i(k+1) = x_i(k) - (\nabla^2 f_i(x_i(k)))^{-1} \nabla f_i(x_i(k)),$$

where $k \in N_+$ is a discrete time index, $x_i(k)$ is the k-th decision variable of node i. The second term on the right is named the local Newton gradient direction of node i.

The iterative formula can be transformed into

$$x_i(k+1) = \psi_i(x_i(k)), \forall i \in \mathcal{V}, \tag{4}$$

where $\psi_i(x_i(k)) = (h_i(x_i(k)))^{-1} g_i(x_i(k))$ is the local Newton-Raphson gradient direction of node i, and

$$h_i(x_i(k)) \doteq \nabla^2 f_i(x_i(k)), \tag{5}$$

$$g_i(x_i(k)) \doteq h_i(x_i(k))x_i(k) - \nabla f_i(x_i(k)). \tag{6}$$

Now we begin to deal with the problem (2), in a global perspective, let $\mathbf{x} = [x_1^T, \cdots, x_N^T]^T$, we need the corresponding global average $\bar{h}(\mathbf{x}(k))$ and $\bar{g}(\mathbf{x}(k))$ which are defined as:

$$\bar{h}(\mathbf{x}(k)) \doteq \frac{1}{N}\sum_{i=1}^{N} \nabla^2 f_i(x_i(k)), \tag{7}$$

$$\bar{g}(\mathbf{x}(k)) \doteq \frac{1}{N}\sum_{i=1}^{N} [h_i(x_i(k))x_i(k) - \nabla f_i(x_i(k))]. \tag{8}$$

and we mark $\psi(\mathbf{x}(k)) \doteq (\bar{h}(\mathbf{x}(k)))^{-1}\bar{g}(\mathbf{x}(k))$.

We consider the following dynamical system

$$x_i(k+1) = \psi(\mathbf{x}(k)), i = 1, \cdots, N. \tag{9}$$

which means every node $i \in \mathcal{V}$ is driven by the same global forcing term $\psi(\mathbf{x}(k))$. Furthermore, under the identical initial condition $x_i(0) = \bar{x}(0), \forall i \in \mathcal{V}$, the trajectories of every node coincide which also means that the constrains $x_i(k) = x_j(k) = \bar{x}(k), \forall i, j \in \mathcal{V}$ are hold in every iteration k.

Then constraints in problem (2) are met, and $\psi(\mathbf{x}(k))$ is the global Newton-Raphson gradient direction of the objective function which means that (9) is the pure Newton iterate of problem (2). When the initial iteration point close to the optimal solution point x^* and $\bar{x}(0) \in S_\delta = \{x \mid \|x - x^*\| \le \delta\}$, under Assumption 2 and Assumption 3, Proposition 1 holds and the conditions in Lemma 1 can be easily satisfied, then the convergence of the iteration (9) is guaranteed.

4.2 Finite-time average consensus

In the undirected and acyclic connected graph $\mathcal{G}$, we seek a peer-to-peer consensus protocol to make the global variables $\bar{h}(\mathbf{x}(k))$ and $\bar{g}(\mathbf{x}(k))$ available for each node through a distributed way in a finite time. In this subsection we modify the algorithm of the finite-time consensus method proposed in [22] to fit our calculations and make it easy to insert into the Newton iteration (9).

Here $l \in N_+$ is used to indicate the number of consensus iterations while k for the Newton ieration. In order to express convenience, for a fixed k in every Newton iteration, let $s_{i\to j}(l)$ and $t_{i\to j}(l)$ denote $g_i(x_i(k))$ and $h_i(x_i(k))$ which were passed from node i to node j at time l without using information from node j. $\omega_{i\to j}(l)$ is the number of nodes used to compute $s_{i\to j}(l)$ and $t_{i\to j}(l)$ and passed from node i to node j at time l. Three internal variables $\tilde{\omega}_i(l)$, $\tilde{s}_i(l)$ and $\tilde{t}_i(l)$ are defined to represent the sum value of the weight and variables in node i in the l-th iteration. We make $\omega_i(l) = \omega_i(k) = 1, \forall i \in \mathcal{V}, \forall l, k > 0$ for the standard average consensus. With the necessary initial conditions, The concrete implementation is summarized below:

1. Initialization step ($l = 0$)

$$\omega_{i\to j}(0) = \omega_i(k), \tag{10}$$

$$s_{i\to j}(0) = g_i(x_i(k)), \tag{11}$$

$$t_{i\to j}(0) = h_i(x_i(k)). \tag{12}$$

2. At iteration $l = 1, 2, \cdots, d$, for each node i, compute

$$\tilde{\omega}_i(l) = \omega_i(k) + \sum_{j\in\mathcal{N}_i} \omega_{j\to i}(l-1), \tag{13}$$

$$\tilde{s}_i(l) = g_i(x_i(k)) + \sum_{j\in\mathcal{N}_i} s_{j\to i}(l-1), \tag{14}$$

$$\tilde{t}_i(l) = h_i(x_i(k)) + \sum_{j\in\mathcal{N}_i} t_{j\to i}(l-1). \tag{15}$$

3. Then for each $j \in \mathcal{N}_i$, compute

$$\omega_{i\to j}(l) = \tilde{\omega}_i(l) - \omega_{j\to i}(l-1), \tag{16}$$

$$s_{i\to j}(l) = \tilde{s}_i(l) - s_{j\to i}(l-1), \tag{17}$$

$$t_{i\to j}(l) = \tilde{t}_i(l) - t_{j\to i}(l-1). \tag{18}$$

Lemma 2. We consider the graph $\mathcal{G}$ is undirected and acyclic with diameter d. Let $\mathcal{V}_i(l)$ is the set of nodes in $\mathcal{V}$ that are at most l hops away from node i ($\forall l > 0$ and i is not in $\mathcal{V}_i(l)$). Naturally we can easily get

$$\tilde{\omega}_i(l) = \omega_i + \sum_{j\in\mathcal{V}_i(l)} \omega_j, \tag{19}$$

$$\tilde{s}_i(l) = g_i(x_i(k)) + \sum_{j\in\mathcal{V}_i(l)} g_j(x_j(k)), \tag{20}$$

$$\tilde{t}_i(l) = h_i(x_i(k)) + \sum_{j\in\mathcal{V}_i(l)} h_j(x_j(k)). \tag{21}$$

Then for every $i \in V$, when $l \ge d$, the average consensus is achieved, that is,

$$\bar{g}(\mathbf{x}(k)) = \frac{\tilde{s}_i(l)}{\tilde{\omega}_i(l)}, \tag{22}$$

$$\bar{h}(\mathbf{x}(k)) = \frac{\tilde{t}_i(l)}{\tilde{\omega}_i(l)}. \tag{23}$$

The proof of Lemma 2 is shown in the appendix. For an arbitrary fixed k, the global variables $\bar{h}(\mathbf{x}(k))$ and $\bar{g}(\mathbf{x}(k))$ can be available for each node $i \in \mathcal{V}$ through the lemma 2. Then an accurate global Newton-Raphson gradient direction $\psi(\mathbf{x}(k)) = (\bar{h}(\mathbf{x}(k)))^{-1}\bar{g}(\mathbf{x}(k))$ is built by a distributed way and available for each node.

4.3 Fast newton consensus algorithm

In this section, we synthesize the Newton iteration algorithm and the finite-time consensus algorithm to build a new fast Newton consensus optimization algorithm (FNC), which is summarized in Algorithm 1.

Algorithm 1 Fast Newton Consensus Algorithm(FNC)

```
Storage allocation and parameters: $x_i, y_i, z_i \in R$, $\omega_i = 1$, $\alpha_i(k) = 1$ for $i \in \mathcal{V}$; $\epsilon(k) = 1, \forall k$, $\eta = m^2/L$
Initialization: $x_i(0)$ for all $i \in \mathcal{V}$
Main loop:
 1: for $k = 0, 1, 2, \cdots$ do
 2:     Inner Iteration Initialization:
 3:     For each nodes $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$,
 4:     $\omega_{i\to j}(0) = \omega_i$
 5:     $s_{i\to j}(0) = g_i(x_i(k))$
 6:     $t_{i\to j}(0) = h_i(x_i(k))$
 7:     for $l = 1, 2, \cdots, d$ do
 8:         for each node $i$, compute
 9:         $\tilde{\omega}_i(l) = \omega_i + \sum_{j\in\mathcal{N}_i} \omega_{j\to i}(l-1)$
10:         $\tilde{s}_i(l) = g_i(x_i(k)) + \sum_{j\in\mathcal{N}_i} s_{j\to i}(l-1)$
11:         $\tilde{t}_i(l) = h_i(x_i(k)) + \sum_{j\in\mathcal{N}_i} t_{j\to i}(l-1)$
12:         then for each $j \in \mathcal{N}_i$, compute
13:         $\omega_{i\to j}(l) = \tilde{\omega}_i(l) - \omega_{j\to i}(l-1)$
14:         $s_{i\to j}(l) = \tilde{s}_i(l) - s_{j\to i}(l-1)$
15:         $t_{i\to j}(l) = \tilde{t}_i(l) - t_{j\to i}(l-1)$
16:     end for
17:     $y(k) = \frac{\tilde{s}_i(d)}{\tilde{\omega}_i(d)}$
18:     $z(k) = \frac{\tilde{t}_i(d)}{\tilde{\omega}_i(d)}$
19:     if $\|y(k)\| \ge \eta$ then
20:         $\epsilon(k) = (k+1)/(k+2)$
21:     end if
22:     for $i = 1, 2, \cdots, N$ do
23:         $x_i(k+1) = (1-\epsilon(k))x_i(k) + \epsilon(k)(z(k))^{-1}y(k))$
24:     end for
25: end for
26: return
```

The outer layer (Steps 19 ~ 24) of the algorithm is a modified Newton iteration for optimization. The inner layer (Steps 2 ~ 18) applies the finite-time average consensus to obtain accurate global decision variables for every node in a finite number of iterations.The main descent iteration is implemented in Step 23, which is a linear combination of the old estimate and the new estimate of the node for the optimal value point x^*. When $\epsilon = 1$, the iteration become the pure Newton iteration. Relative to (9), the additional parameters $\epsilon \in (0, 1]$ and $(1-\epsilon)$ are a conservative mechanism.

Under the same initial value $\mathbf{x}(0) = x(0) \otimes I_N$ and $x(0) \in S_\delta$, the main descent iteration in Step 23 is

$$\begin{aligned}
&x_i(k+1)\\
&= (1-\epsilon)x_i(k) + \epsilon(z(k))^{-1}y(k))\\
&= (1-\epsilon)x_i(k) + \epsilon(\bar{h}(\mathbf{x}(k)))^{-1}\bar{g}(\mathbf{x}(k))\\
&= x_i(k)\\
&\quad - \epsilon\left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(x_i(k))\right)^{-1}\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(x_i(k)).
\end{aligned}$$

The iterative process is the same for any $i \in \mathcal{V}$, that is, $x_i(k) = x_j(k) = \hat{x}(k), \forall i, j \in \mathcal{V}$ and $\forall k > 0$. Then iteration of each node can be uniformly written as follows:

$$x(k+1) = x(k) - \epsilon(k)\nabla^2 f(x(k))^{-1}\nabla f(x(k)). \tag{24}$$

Through the above simplification, $\epsilon(k)$ can also be seen as a common iteration step size. In the following theorem we show the convergence property of Algorithm 1.

Theorem 1. Consider the dynamics in Algorithm 1 with Assumptions 1, 2 and 3. Taking $x_i(0) = x(0), \forall i \in \mathcal{V}$, with $x(0) \in dom(f)$, the sequence $\{x_i(k)\}, \forall i \in \mathcal{V}$, generated by the algorithm has the following properties: For $\eta = m^2/L$,

1. If $\|\nabla f(x(k))\| \ge \eta$, then $f(x(k)) - f(x^*) \le -\gamma$, where $\gamma = \alpha\beta\eta^2\frac{m}{M^2}$.
2. If $\|\nabla f(x(k))\| < \eta$, then $\{\|f(x(k)) - f(x^*)\|\}$ decreases at a quadratic rate. Furthermore, when $L/2m < 1$, the sequence $\{\|x(k) - x^*\|\}$ is also quadratic convergent.

This theorem ensures the convergence of the algorithm for $x(0) \in$ **dom**f. It eventually has a quadratic convergence rate and trajectories of every node are consistent with the standard Newton iteration. The detailed proof is in the appendix.

Remark 3. Although the initial value can be arbitrarily selected within the domain. However, a bad initial value (away from the optimal value point) will increase the iteration number. And in the general case where the initial condition in every node i is not identical, we need to add a finite-time consensus process for $x_i(0)$ in the whole network before the optimization process. In this sense, the initial value of each node can be chosen as the optimal point of the respective objective function. This will significantly reduce the number of iterative steps.

V. COMPARISON ANALYSIS AND NUMERICAL SIMULATION

In this subsection, we study the performance of our algorithm and compare with two representative methods in a tree communication network environment. The topological network is showed in Fig. 1. For convenience of presentation and simplification of calculation, we limit the variable $x(k)$ to scalar form and the scalar cost function with the form $f_i(x) = c_i e^{a_i x} + d_i e^{-b_i x}$, $i = 1, 2, \cdots, N$, with $a_i, b_i \sim U[-1, 1]$, $c_i, d_i \sim U[0, 1]$ where U indicates the uniform distribution. These f_i and f fulfill Assumption 2 and 3. Fig. 2 shows images of these f_i and f and indicates the optimal value point $x^* = 0.3275$ (compute in advance). The relative MSE (mean square error) is defined:

$$MSE(k) = \frac{1}{N}\sum_{i=1}^{N} \frac{\| x_i(k) - x^* \|^2}{\| x^* \|^2}. \tag{25}$$

5.1 Comparison with existing methods

In this subsection, we compare the performance of the proposed Fast Newton Consensus (FNC) method with the Newton-Raphson Consensus (NRC) method in [14], and Distributed Sub-gradient Methods (DSM) in [6]. The mean of the total nodes in the k-th iteration is expressed as $x^{-}(k)$. Taking the initial value $x_i(0) = 5$. In NRC, based on the above analysis, let $\varepsilon = 0.1$, $P = I_N - (1/4) * L$ is the max degree weight matrix which is the doubly stochastic matrix. In DSM, we use the same weight matrix P and choose the step size $\epsilon(k) = \rho/k$ and $\rho > 0$.

Fig. 3 shows the performance of the three methods. In the first panel of Fig. 3, the convergence performance of the proposed algorithm is illustrated. The update trajectory of each node coincides. The convergence performance of NRC are shown in the second panel of Fig. 3, In the last panel of Fig. 3, in contrast, the DSM has the worst performance. In general, our

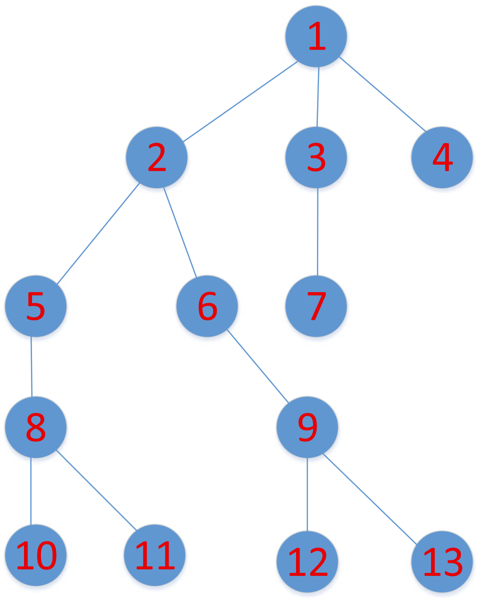

Fig. 1. Network topology.

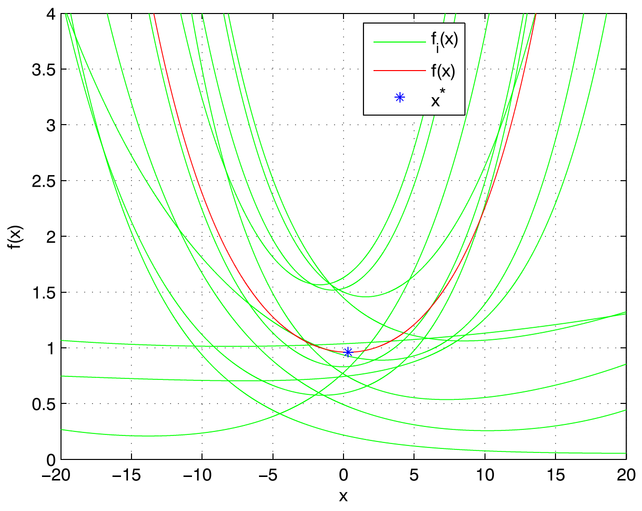

Fig. 2. Graphs of the functions f_i and f,together with the optimal value point $x^* = 0.3275$.

algorithm has the best aggregation effect and the fastest convergence speed. Fig. 4 compares the mean evolution of those iterative processes based on the relative MSE. The convergence of proposed algorithm is extremely rapid. Once the second condition in the theorem ise satisfied, the rate of quadratic convergence and trajectories of every node are consistent with the standard Newton iteration, which means that the algorithm happens to be

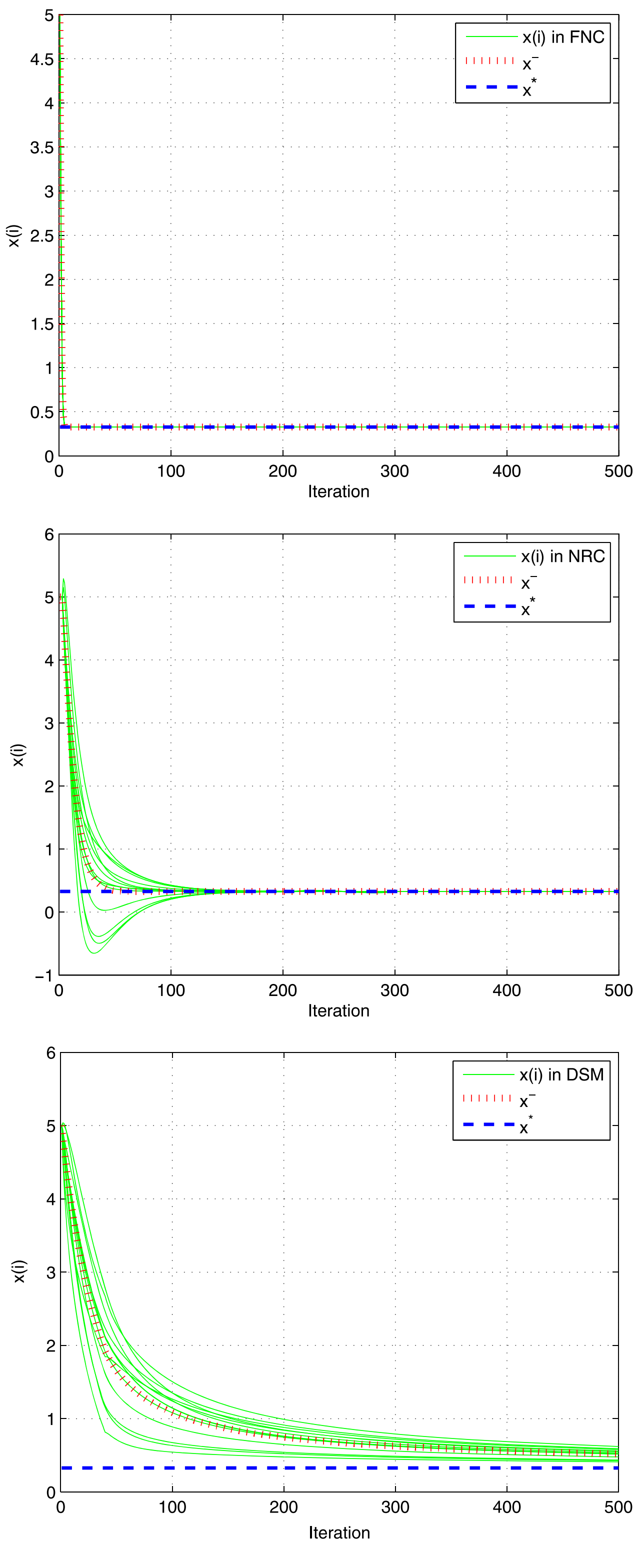

Fig. 3. Fast Newton Consensus (FNC), Newton-Raphson Consensus (NRC) and Distributed Sub-gradient Methods (DSM) with $\mathbf{x}(0) = 5$.

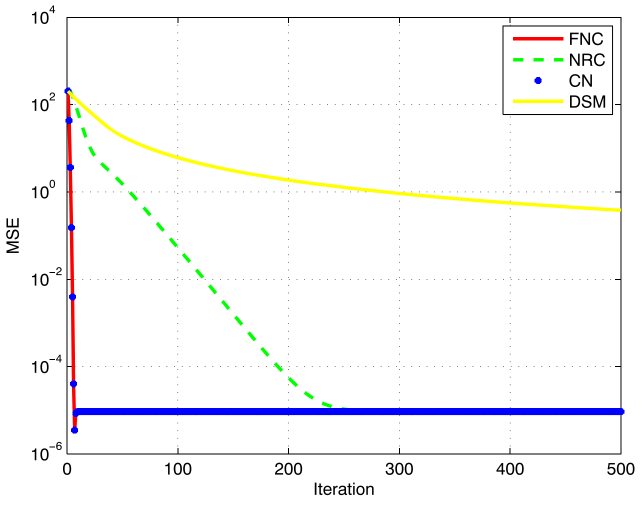

Fig. 4. MSE of the three algorithms together with Central Newton method (CN) in $\mathbf{x}(0) = 5$.

a distributed implementation of the standard centered Newton method.

5.2 Complexity analysis and comparison

The design principle of our algorithm is to reduce the number of outer iterations at the expense of adding multi-step inline finite-time consensus, while algorithm NRC only includes a single common consensus step. Every single iteration of the two consensus involves roughly the same amount of information exchange and computation.

In Fig. 4, we truncate our algorithm within $k = 8$ iterations and ensure good convergence accuracy (10^{-5}). In order to fairly compare the two algorithms, under the same amount of information exchange, we compare the performance of our algorithm in 8-th iteration and the NRC in $48(8 \times 6)$-th iteration where $d = 6$, our algorithm enjoys an outstanding convergence accuracy. With the same accuracy requirements (10^{-5}), The ratio of external iterations is almost 10 to 200. The computation and memory costs of our algorithm are far less than the NRC With a rough ratio of 60 to 200.

VI. CONCLUSION

We have presented a novel fast distributed Newton algorithm to solve a convex optimization problem. Under necessary assumptions, our algorithm is a distributed implementation of standard Newton iterations, and a quadratic linear convergence rate can be guaranteed through the theoretical analysis and numerical simulation. Compared with other representative first-order and second-order distributed gradient optimization methods, the simulation results further demonstrate the advantages of the algorithm in convergence speed and efficiency. As future work, we will try to extend the proposed algorithm to loop graph and consider the convex optimization problem with local and global constraints.

VII. APPENDIX

Proof of Lemma 2. The concrete proof is taken from the literature [22]. As the graph is acyclic, the decomposition $\mathcal{G} = \mathcal{G}_i \cup (i,j) \cup \mathcal{G}_j$ can be done simply by only removing the edge (i,j) where $\mathcal{G}_i$ and $\mathcal{G}_j$ are the disjoint subgraphs of $\mathcal{G}$. $\mathcal{V}_i$ and $\mathcal{V}_j$ are the corresponding sets of nodes for disjoint subgraphs respectively.

At the initialization step ($l = 0$), $\omega_{i\to j}(0) = \omega_i(k) = 1$. According to the previous relevant definitions, $\omega_{i\to j}(1)$ contains all $\omega_{m\to i}(0)$ for all nodes m in $\mathcal{N}_i$, except node j, and with the definition of $\mathcal{V}_i(1)$, this calculation can be expressed as

$$\omega_{i\to j}(1) = \omega_i + \sum_{m\in\mathcal{V}_i(1)\setminus\mathcal{V}_j} \omega_m. \tag{26}$$

For the case $l = 2$, $\omega_{i\to j}(2)$ contains all $\omega_{m\to i}(1)$ for all nodes m in $\mathcal{N}_i$ except node j. In other words $\omega_{i\to j}(2)$ includes all the nodes' weights at least 2 hops away from node i, except those in $\mathcal{V}_j$, that is

$$\omega_{i\to j}(2) = \omega_i + \sum_{m\in\mathcal{V}_i(2)\setminus\mathcal{V}_j} \omega_m. \tag{27}$$

Repeat the above, through mathematical induction, we can get,

$$\omega_{i\to j}(l) = \omega_i + \sum_{m\in\mathcal{V}_i(l)\setminus\mathcal{V}_j} \omega_m, \forall l > 0. \tag{28}$$

If node $j (\in \mathcal{N}_i)$ is l hops away from node i, then all the nodes in $\mathcal{G}_j$ that are $l-1$ hops away from node j are actually l hops away from node i, that is,

$$\mathcal{V}_i(k) \setminus \mathcal{V}_j \sqcup \mathcal{V}_j(k-1) \setminus \mathcal{V}_i \sqcup \{j\} = \mathcal{V}_i(k). \tag{29}$$

Then we naturally get the following form:

$$\begin{aligned}\tilde{\omega}_i(l) &= \omega_{i\to j}(l) + \omega_{j\to i}(l-1) \\ &= \omega_i + \sum_{m\in\mathcal{V}_i(l)\setminus\mathcal{V}_j} \omega_m + \omega_j + \sum_{m\in\mathcal{V}_j(l-1)\setminus\mathcal{V}_i} \omega_m \\ &= \omega_i + \sum_{m\in\mathcal{V}_i(l)} \omega_m.\end{aligned} \tag{30}$$

And the other two similar form of $\tilde{s}_i(l)$ and $\tilde{t}_i(l)$ in the lemma can be shown in the same way.

When $l \geq d$, $\mathcal{V}_i(l) = \mathcal{V}, \forall i \in \mathcal{V}$. With the above form, the internal sum variables $\tilde{w}_i(l)$, $\tilde{s}_i(l)$ and $\tilde{t}_i(l)$ will remain unchanged for every i and the corresponding values are the sum of the corresponding state values of the entire network node without duplication and omission. The average consensus are achieved by corresponding division (22-23).

□

Proof of Theorem 1. As the iterative process of each node can be uniformly written as follows:

$$x(k+1) = x(k) - \epsilon(k)\nabla^2 f(x(k))^{-1}\nabla f(x(k)). \tag{31}$$

Similar to standard Newton convergence analysis in [4], The proof is divided into two phases: the damped Newton phase and the pure Newton phase.

(1) Assume $\|\nabla f\| \geq \eta$, which means the initial point is far from the optimal value point. We firstly show there will be a lower bound on the step size. We note $\Delta x_{nk} = -(\nabla^2 f(x(k)))^{-1}\nabla f(x(k))$, then $x(k+1) = x(k) + \epsilon(k)\Delta x_{nk}$. The bound $mI \leq \nabla^2 f(x) \leq MI$ implies an upper bound on the function $f(k+1)$:

$$\begin{aligned}f(x(k+1)) \leq& f(x(k)) + \epsilon(k)\nabla f(x(k))^T \Delta x_{nk} \\ &+ \frac{M\|\Delta x_{nk}\|^2 t^2}{2} \\ \leq& f(x(k)) - \epsilon(k)\lambda(x(k))^2 \\ &+ \frac{M}{2m}\epsilon(k)^2\lambda(x(k))^2,\end{aligned} \tag{32}$$

where $\lambda(x) := (\Delta x_{nk}^T \nabla^2 f(x(k))\Delta x_{nk})^{1/2}$ and $\lambda(x)^2 \geq m\|\Delta x_{nk}\|^2$. Minimize the right part of the above inequality for $\epsilon(k)$:

$$\begin{aligned}f(x(k) + \hat{\epsilon}(k)\Delta x_{nk}) \leq& f(x(k)) - \frac{m}{2M}\lambda(x(k))^2 \\ \leq& f(x(k)) - \alpha\epsilon(k)\lambda(x(k))^2.\end{aligned}$$

Here $\hat{\epsilon}(k) = m/M$ and it satisfies the exit condition of the line search where $\alpha \in (0, 1/2]$ and $\beta \in (0, 1/2)$. And then $\epsilon(k) = (k+1)/(k+2)$ is feasible as an actual calculation with $\epsilon(k) \geq \beta m/M, \forall k \geq 1$. As $\lambda(x)^2 \geq (1/M)\|\nabla f(x(k))\|^2$ and we noted $\gamma = \alpha\beta\eta^2\frac{m}{M^2}$ then

$$\begin{aligned} f(x(k)) - f(x(k+1)) &\geq \alpha\epsilon(k)\lambda(x(k))^2 \\ &\geq \alpha\beta\frac{m}{M}\lambda(x(k))^2 \\ &\geq \alpha\beta\frac{m}{M^2}\|\nabla f(x(k))\|^2 \\ &\geq \alpha\beta\eta^2\frac{m}{M^2}. \end{aligned} \tag{33}$$

This means that our iteration not only ensures that the value of the function continues to fall, but also that the magnitude of the drop has a corresponding lower bound.

(2) If $\|\nabla f(x(k))\| \leq \eta$, which means that the iteration variable $x(k)$ enters a small δ domain of the optimal value point. We can choose $\epsilon(l) = 1, \forall l \geq k$ and

$$\frac{L}{2m^2}\|\nabla f(x(k+1))\| \leq (\frac{L}{2m^2}\|\nabla f(x(k))\|)^2. \tag{34}$$

A detailed explanation of above conclusion can be found in [4]. Applying the above inequality recursively, for $l \geq k$,

$$\frac{L}{2m^2}\|\nabla f(x(l))\| \leq (\frac{L}{2m^2}\|\nabla f(x(k))\|)^{2^{l-k}} \leq (\frac{1}{2})^{2^{l-k}},$$

and hence

$$f(x(l)) - f(x^*) \leq \frac{1}{2m}\|\nabla f(x(l))\|^2 \leq \frac{2m^3}{L^2}(\frac{1}{2})^{2^{l-k+1}}.$$

This inequality shows that convergence is quadratic convergence which is extremely rapid. To further analyze the iteration of x where $x_i = x, \forall i \in \mathcal{V}$:

$$\begin{aligned} & x(k+1) - x^* \\ &= x(k) - x^* \\ &\quad - \epsilon(k)(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(x(k)))^{-1}\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(x(k)) \\ &= (\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(x(k)))^{-1}\{\frac{1}{N}\sum_{i=1}^{N}\int_0^1[\nabla^2 f_i(x(k)) \\ &\quad - \epsilon(k)\nabla^2 f_i(x^* + t(x(k) - x^*))]dt(x(k) - x^*)\} \\ &\leq \frac{1}{m}\frac{1}{N}\sum_{i=1}^{N}\int_0^1[\nabla^2 f_i(x(k)) - \epsilon(k)\nabla^2 f_i(x^* \\ &\quad + t(x(k) - x^*))]dt(x(k) - x^*). \end{aligned} \tag{35}$$

Let $\epsilon(l) = 1, \forall l \geq k$, (35) can be further relaxed as

$$\begin{aligned} & \|x(k+1) - x^*\| \\ &\leq \frac{1}{m}\{\frac{1}{N}\sum_{i=1}^{N}\int_0^1[Lt\epsilon\|x(k) - x^*\|]dt(x(k) - x^*)\} \\ &\leq \frac{L\epsilon}{2m}\|x(k) - x^*\|^2. \end{aligned}$$

When $\frac{L}{2m} < 1$, the sequence $\{\|x(k+1) - x^*\|\}$ is at least quadratic linear convergence. □

REFERENCES

1. Johansson, B., "On distributed optimization in networked systems," *PhD dissertation*, Stockholm (2008).
2. Zhu, S., C. Chen, W. Li, B. Yang, and X. Guan, "Distributed optimal consensus filter for target tracking in heterogeneous sensor networks," *IEEE T. Cybern.*, Vol. 43, No. 6, pp. 1963–1976 (2013).
3. Predd, J. B., S. R. Kulkarni, and H. V. Poor, "A collaborative training algorithm for distributed learning," *IEEE Trans. Inf. Theory*, Vol. 55, No. 4, pp. 1856–1871 (2009).
4. Boyd, S. and L. Vandenberghe, *Convex Optimization*, 1st edition, Cambridge University Press, Cambridge,United Kingdom (2004).
5. Bertsekas, D. P., *Nonlinear Programming*, 2nd edition, Athena Scientific, Belmont, Massachusetts (1999).
6. Nedic, A. and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, Vol. 54, No. 1, pp. 48–61 (2009).
7. Nemirovsky, A. S. and D. B. Yudin, *Problem Complexity and Method Efficiency in Optimization*, John Wiley, New York (1983).
8. Nesterov, Y., "A method of solving a convex programming problem with convergence rate $o(1/k^2)$," *Sov. Math. Dokl.*, Vol. 27, pp. 372–376 (1983).
9. Nesterov, Y., *Introductory Lectures on Convex Optimization: A Basic Course*, 1st edition, Springer, US (2004).
10. Ghadimi, E., H. R. Feyzmahdavian, and M. Johansson, "Global convergence of the heavy-ball method for convex optimization," *2015 Eur. Control Conf. (ECC)*, Linz, Austria, pp. 310–315 (2015).
11. Qu, G. and N. Li, "Accelerated distributed nesterov gradient descent for smooth and strongly convex functions," *54th IEEE Annual Allerton Conference Commun., Control, Comput.*, Illinois, USA, pp. 209–216 (2017).

12. Wei, E., A. Ozdaglar, and A. Jadbabaie, "A distributed Newton method for network utility maximization;Part I: Algorithm," *IEEE Trans. Autom. Control*, Vol. 58, No. 9, pp. 2162–2175 (2014).
13. Mokhtari, A., Q. Ling, and A. Ribeiro, "Network newton distributed optimization methods," *IEEE Trans. Signal Process.*, Vol. 65, No. 1, pp. 146–161 (2016).
14. Varagnolo, D., F. Zanella, A. Cenedese, G. Pillonetto, and L. Schenato, "Newton-raphson consensus for distributed convex optimization," *IEEE Trans. Autom. Control*, Vol. 61, No. 4, pp. 994–1009 (2016).
15. Olfati-Saber, R., J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, Vol. 95, No. 1, pp. 215–233 (2007).
16. Hou, W., M. Fu, and H. Zhang, "Distributed consensus of third-order multi-agent systems with communication delay: Consensus of third-order multi-agent systems," *Asian J. Control*, Vol. 20, No. 9, pp. 956–961 (2017).
17. Ko, C. K. and X. Gao, "On matrix factorization and finite-time average-consensus," *Decision and Control*, Shanghai, China, pp. 5798–5803 (2009).
18. Oliva, G., R. Setola, and C. N. Hadjicostis, "Distributed finite-time average-consensus with limited computational and storage capability," *IEEE Trans. Control Netw. Syst.*, Vol. 4, No. 2, pp. 380–391 (2017).
19. Wang, F., X. Chen, Y. He, and M. Wu, "Finite-time consensus problem for second-order multi-agent systems under switching topologies," *Asian J. Control*, Vol. 19, No. 5, pp. 1756–1766 (2017).
20. Wang, Q., Y. Wang, and C. Sun, "Fixed-time consensus of multi-agent systems with directed and intermittent communications," *Asian J. Control*, Vol. 19, No. 1, pp. 95–105 (2017).
21. Ning, B., Z. Zuo, J. Jin, and J. Zheng, "Distributed fixed-time coordinated tracking for nonlinear multi-agent systems under directed graphs," *Asian J. Control*, Vol. 20, No. 2, pp. 646–658 (2018).
22. Xie, K. and M. Fu, "A new finite-time distributed algorithm for weighted average consensus (2016)." submitted for publication.
23. Makki, S. A. M. and G. Havas, "Distributed algorithms for depth-first search," *Inf. Process. Lett.*, Vol. 60, No. 1, pp. 7–12 (1996).

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 has been pursuing his Ph.D. degree at the School of Control Science and Engineering, Shandong University, China. His research interests include multi-agent network consensus control and distributed optimization.

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. From 1987 to 1989, he served as an Assistant Professor in the Department of Electrical and Computer Engineering, Wayne State University, Detroit, Michigan. 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, IEEE Transactions on Signal Processing and Journal of Optimization and Engineering.

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.

Peng Cui received the M.S. degrees in science computing in 1999 and the Ph.D. degree in control theory and control engineering from Shandong University, P.R. China, in 2008. He joined Shandong University in 2008 and his current research interests include distributed algorithm, state estimation, optimal control and networked control systems.

Zhipeng Li received his B.S. degree and M.S. degree in mathematics from Jining University in 2011 and Shandong University, Jinan, China, in 2014, respectively. Since 2014 he is pursuing his Ph.D. degree at the School of Control Science and Engineering, Shandong University. His research interests include stochastic systems, distributed estimation and optimization.