DOI: 10.3969/j. issn. 1673-5188. 2017. 03. 005
http://kns.cnki.net/kcms/detail/34.1294.TN.20170712.1425.002.html, published online July 12, 2017

Distributed Least-Squares Iterative Methods in Large-Scale Networks: A Survey

SHI Lei[1], ZHAO Liang[2], SONG Wenzhan[3], Goutham Kamath[1], WU Yuan[4], and LIU Xuefeng[5]

(1. Georgia State University, Atlanta, GA 30302, USA;
2. Georgia Gwinnett College, Lawrenceville, GA 30043, USA;
3. University of Georgia, Athens, GA 30602, USA;
4. Zhejiang University of Technology, Hangzhou 310023, China;
5. The Hong Kong Polytechnic University, Hong Kong, China)

Abstract

Many science and engineering applications involve solving a linear least-squares system formed from some field measurements. In the distributed cyber-physical systems (CPS), each sensor node used for measurement often only knows partial independent rows of the least-squares system. To solve the least-squares all the measurements must be gathered at a centralized location and then perform the computation. Such data collection and computation are inefficient because of bandwidth and time constraints and sometimes are infeasible because of data privacy concerns. Iterative methods are natural candidates for solving the aforementioned problem and there are many studies regarding this. However, most of the proposed solutions are related to centralized/parallel computations while only a few have the potential to be applied in distributed networks. Thus distributed computations are strongly preferred or demanded in many of the real world applications, e.g. smart-grid, target tracking, etc. This paper surveys the representative iterative methods for distributed least-squares in networks.

Keywords

distributed computing; iterative methods; least-squares; mesh network

1 Introduction

Many physical phenomena can be described by partial differential equations [1] which forms large sparse system of linear equations after further discretized. Problems, such as state estimation, target tracking and tomography inversion, are often formulated as a large-scale linear system based on some field measurements. Those field measurements may contain errors, thus an extra amount of measurement is often sampled to form an over-determined linear system:

$$Ax \approx b, \tag{1}$$

where $A \in \mathbb{R}^{m \times n} (m \geqslant n)$, $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. Such extra information can smooth out the errors but produces an overdetermined system that usually has no exact solution. The method of least-squares is a common solution to the above problem and can be defined as

$$\min_x \| Ax - b \|_2 . \tag{2}$$

The coefficient A is often modeled from the data obtain from sensors used for observing the physical phenomena such as cyber physical system. Each sensor or node observes partial phenomena due to the spatial and temporal restriction and thus only forms partial rows of the least-squares systems. The large-scale cyber-physical systems are often built on a mesh network, which could be a wired, wireless or wired-wireless hybrid multi-hop network. For instance, the problems from target tracking, seismic tomography and smart grid state estimation all have an inherently distributed system of linear equations. However, the least squares method used currently for solving these problems assumes a centralized setup, where partial row information from all the nodes are collected in a server and then solved using the centralized least-square algorithm.

In many of those cyber-physical systems, the distributed computation in mesh networks is strongly demanded or preferred over the centralized computation approach, due to the following reasons (but not limited to):

1) In some applications such as imaging seismic tomography with the aid of mesh network, the real-time data retrieval from a large-scale seismic mesh network into a central server is virtually impossible due to the sheer amount of data and resource limitations. The distributed computation may process data inside the network in real time to reduce the bandwidth demand as well as distribute the communication and computation load to each node in the network.
2) The mesh network may be disruptive in real world and the data collection and centralized computation may suffer from node failure or link disruption. These become a bottleneck especially when the node failure or link disruption happens

This work is partially supported by US NSF under Grant No. NSF-CNS-1066391 and No. NSF-CNS-0914371, NSF-CPS-1135814 and NSF-CDI-1125165.

near to the sink node which leads to loss of high volume of raw data. However, with distributed computation, the remaining nodes in the network can finish the computation and get the approximated results.

3) In smart grid state estimation, the data collection for centralized computation is even infeasible due to data privacy concerns or inter-agency policy constraints.

4) In some applications that need real-time control, the distributed computation also has advantage over centralized schemes, since some decisions can be made locally in real time. The current state of the art computational device such as smart phones enables us to perform in-network computing and carry out distributed computation over a mesh network.

Iterative methods are natural candidates when it comes for large sparse system and especially for distributed computation of least-squares. Although there are a lot of studies on iterative least-squares, most of them are concerned with the efficiency of centralized/parallel computation, and only a few are explicitly about distributed computation or have the potential to be applied on mesh networks. In mesh networks, since the computers need to communicate with each other through messages passing over a multi-hop network, the key challenges are speeding up the computation and reducing the communication cost. More attention shall be paid to communication instead of the computation cost, especially when solving a big problem in a large-scale mesh network.

In this paper, we select and survey the representative iterative methods from several research communities. These methods have the potential to be used in solving least squares problem over mesh networks. Here, a skeleton sketch of each algorithm is provided and later we analyze the time-to-completion and communication cost of these algorithms and provide the comparison. Some of the algorithms presented here were not originally designed for meeting our requirements, so we slightly modify them to maintain consistency.

The rest of the paper is organized as follows. In section 2, we present the network model and the evaluation criteria for comparison. Then in section 3, we describe the state of art on surveyed algorithms in details, analyze and compare their communication costs and time-to-completion. Finally, we conclude the paper in section 4.

2 Model and Assumption

We denote a wired and/or wireless mesh network with N nodes $v_1, ..., v_N$ which form connected graph and can be reached through multi-hop message relays. Without loss of generality, we assume that the diameter of the network is $\log N$ (i.e., any message can be sent from one node to another through at most $\log N$ hops). We also assume that each node has a single radio and the link between the neighboring nodes has a unit bandwidth. Therefore, the communication delay of one unit data delivery between direct neighbors (either through a unicast to one direct neighbor or multicast/broadcast to all direct neighbors) would be one unit time. It is noted that the link layer supports broadcast, which is often true in many mesh networks. If the link layer only supports unicast, the analysis can be similarly done by considering a one-hop broadcast as multiple unicast and we skip this analysis as it is trivial. We also understand that the link layer communication may take more than one unit time for one unit data due to network interference and media contentions. Therefore, we classify the communication patterns in a mesh network into three categories, unicast (one-hop or multi-hop), one-hop broadcast (local broadcast to all neighbors) and network flooding (broadcast to all the nodes in network). For simplicity and convenience, in the rest of the paper we use the term "broadcast" for local broadcast to one-hop neighbors and "flood" for network flooding. We use the aforementioned assumption on communication cost and delay for the quest of the fundamental limit of each surveyed algorithm in an ideal mesh network.

We assume a random communication network (has to be a connected graph) in our analysis. The influence of the communication network to the performance has been studied in [2], [3]. It is known that the network connectivity ratio, which is defined as the number of edges divided by the number of all possible edgeswill affect the performance. The algorithms are supposed to obtain a faster convergence speed when each node has higher number of neighbors. Higher neighbor count means more nodes can receive the information after one transmission of certain node. It accelerates information diffusion among all the nodes in the network and thus help all the nodes reach consensus faster. To conduct a fair comparison, we use the same communication network (topology) for all the benchmarks.

For comparison and evaluation, the following two performance criteria are considered:

- Communication cost: To solve a least-squares problem of large-size system, communication cost has a big influence on the algorithm performance. Here we refer the communication cost as the cost involved in the messages exchanged in the mesh network during a single iteration of the iterative methods. Since iterative methods typically converge after many iterations, the communication cost of the iterative methods depends on both the cost in one iteration and the iteration number.
- Time-to-completion: The time taken for a network to finish one iteration in the iterative method is referred as time-to-completion in this paper. Note that it is different from the computational time complexity and shall include the consideration of the message size and number of hops the packet has traversed.

We also focus on the analysis of communication delays for time-to-completion while ignoring the computation time in each node. The rationale is summarized as follows. First, the cost of communication is very high in practice. We are highly

constrained by the physical bandwidth and energy consumption in large-scale sensor networks in particular. Furthermore, the communication stage is much more time-consuming than the computation step. It turns out that reducing the total communication rounds is the key for speeding up the algorithm in terms of time-to-completion.

The least-squares problem in (2) formed over the mesh network is inherently distributed, i.e. each node v_u only knows part of A and b. We assume that each node in the network holds $m_u = m/N$ consecutive rows of matrix $A \in \mathbb{R}^{m \times n}$ and the corresponding part of vector b. For example in **Fig. 1**, block A_1 indicates the first m/N rows of matrix A which is assigned to node v_1 along with the right hand side vector $b = \{b_1, \cdots, b_{m/N}\}$. Note that the algorithm surveyed in this paper does not require that matrix A and b be equally partitioned over the network. Here the assumption of equal partition is for the simplicity of presentation and analysis and the new distributed equation takes the form:

$$Ax = b, \tag{3}$$

where,

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_N \end{pmatrix}; b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}; A_u \in \mathbb{R}^{m_u \times n}; b_u \in \mathbb{R}^{m_u}. \tag{4}$$

The least squares problem takes the form $\min_x \|Ax - b\|_2$ and since there is no central coordinator which has entire A and b, the computation of optimum x has to be done distributedly. As mentioned above, the communication cost becomes crucial for distributed solution over sensor network and the goal of this paper is to survey various distributed least squares algorithm originating from different domains. We also try to compare different algorithms under similar criteria as mentioned above so that it provides the reader some basic differences between them and also help them to choose the type of algorithms suitable for their application.

Indices	1	2	3	⋯	$n-1$	n	Block
1	X	X	X	⋯	X	X	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	A_1
m/N	X	X	X	⋯	X	X	
$m/N+1$	X	X	X	⋯	X	X	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	A_2
$2m/N$	X	X	X	⋯	X	X	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$m-m/N+1$	X	X	X	⋯	X	X	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	A_N
m	X	X	X	⋯	X	X	

▲**Figure 1. Row partition of matrix *A*.**

The notations used in this paper are described in **Table 1**.

3 Survey and Analysis

The methods for solving the linear least-squares problems are typically classified into two categories, direct methods and iterative methods. Direct methods are based on the factorization of the coefficient matrix A into easily invertible matrices whereas iterative methods solve the system by generating a sequence of improving approximate solutions for the problem. Until recently direct methods were often preferred over iterative methods [4] due to their robustness and predictable behaviors (one can estimate the amount of resources required by direct solvers in terms of time and storage) [5], [6]. On the other hand, a number of iterative methods have also been discovered, which require fewer memory and are approaching the solution quality of direct solvers [6]. The size of the least squares problem arising from real world three-dimension problem models could be significantly large comprising hundreds of millions of equations as well as the unknowns. Despite such a huge dimension, the matrices arising are typically sparse and can be easily stored. Now given the dimension and sparsity property of the matrix, iterative methods become almost mandatory for solving them [7]. Moreover, iterative methods are gaining ground because they are easier to be implemented efficiently in high-performance computers than direct methods [6].

The methods for solving least-squares problems in distributed networks can be classified into two categories, distributed and decentralized (fully distributed) methods. We will discuss it in detail in this section.

3.1 Distributed Least-Squares Methods

To achieve high performance in computation, researchers have studied distributed iterative methods to solve large linear systems/linear least-squares problems [8], [9]. The researches leverage both shared and distributed memory architecture. In this section, we only present those distributed iterative methods that can be potentially distributed over a mesh network. The Distributed Multisplitting (D-MS), Distributed Modified

▼**Table 1. List of notations**

Notation	Definition
A, E, L, U, Q, R	Matrices
x, y, a, b, r	Vectors
$\alpha, \beta, \delta, \lambda, \gamma$	Scalars
m, n	Rows and columns of matrices
$\mathbb{R}$	Real space
N	Network size
D_{avg}, D_{max}	Average and maximum node degree in the network
u, v	Nodes in the network
k	Iteration number of iterative methods

Conjugate Gradient Least-Squares (D-MCGLS), Distributed Component-Average Row Projection (D-CARP), Distributed Cooperative Estimation (D-CE), Distributed Least Mean Squares (D-LMS), Distributed Recursive Least-Squares (D-RLS) methodsare discussed and compared, but only D-LMS and D-RLS are analyzed in details in this paper since they are more relevant and more promising than other distributed algorithms.

Table 2 gives a summary of the communication cost and time-to-completion of the selected algorithms running in the distributed network. The details about the algorithm description and analysis are shown in Section 3. When the least-squares problem in (2) where $A\in\mathbb{R}^{m\times n}(m\geqslant n)$, $x\in\mathbb{R}^n$ and $b\in\mathbb{R}^m$ is considered, we suppose that the iterative algorithm converges within k iterations in the network, and D_{avg} and $D_{\max}$ denote the average and maximum node degrees of the network respectively. The algorithms discussed in this section have been proved to be convergent, but the iteration number highly depends on the matrix condition number; these algorithms may need hundreds to thousands of iterations to converge over a network with hundreds of nodes for a large system. Besides, some algorithms either requires flooding communication in the network per iteration or a Hamiltonian path in the network to perform the computation node by node.

3.1.1 Distributed Least Mean Squares Method

Schizas, Mateos and Giannakis [10]–[13] introduce the D-LMS algorithm. This algorithm lets each node maintain its own local estimation and, to reach the consensus, exchange the local estimation only within its neighbors. The advantage of the methods like D-LMS and D-CE in signal processing is that only local information exchange is required. The problem is that these methods may converge slow in a large-scale network.

In their discussion, the wireless sensor network is deployed to estimate a signal vector $x^*\in\mathbb{R}^{n\times 1}$. Each node v_u has a regression vector $A_u(k)\in\mathbb{R}^{n\times 1}$, where $k=0,1,2,\ldots$ denotes the time instants, and there is a observation $b_u(k)$ on time k; both of them are assumed to have zero mean. One global vector $b(k):=[b_1(k)\cdots b_N(k)]^T\in\mathbb{R}^{N\times 1}$ is used for all the observations on N nodes in the network. $A(k):=[A_1(k)\cdots A_N(k)]^T\in\mathbb{R}^{N\times n}$ is the regression vector combined over the network, and the global LMS estimator is then described as

$$\hat{x}(k)=\arg\min_x E[\|b(k)-A(k)x\|^2]=\arg\min_x\sum_{u=1}^{N}E[(b_u(k)-A_u^T(k)x)^2]. \tag{5}$$

Let $\{x_u\}_{u=1}^N\in\mathbb{R}^n$ represent the local estimation of the global variable x of one node v_u (each node has its own estimation of the signal vector). In conjunction with these local variables, we consider the convex constrained minimization problem as

$$\{\hat{x}_u(k)\}_{u=1}^N=\arg\min_x\sum_{u=1}^{N}E[(b_u(k)-A_u^T(k)x_u)^2],\quad s.t.x_u=x_{u'},u\in N,u'\in N_u, \tag{6}$$

where N_u is the neighbor set of node v_u.

The equality constraints above only involve the local estimations of the neighbors of each node and force an agreement among each node's neighbors. Since we assume that the network is connected, the constraints above will introduce a consensus in the network. We can finally have $x_u=x_{u'}$ for all $u,u'\in N$. We find that the distributed estimation problem is equivalent to the original problem in the sense that their optimal solutions coincide such as $\hat{x}_u(k)=\hat{x}(k)$, for all $u\in N$.

To construct the distributed algorithm, the authors resort to the Alternating Direction Method of Multipliers (ADMM) (algorithm, and get the following two equations for estimation updating,

$$v_u^{u'}(k)=v_u^{u'}(k-1)+\frac{c}{2}\left(x_u(k)-\left(x_{u'}(k)+\eta_u^{u'}(k)\right)\right),\quad u'\in N_u, \tag{7}$$

$$x_u(k+1)=x_u(k)+\mu_u[2A_u(k+1)e_u(k+1)-\sum_{u'\in N_u}\left(v_u^{u'}(k)-\left(v_{u'}^{u}(k)+\bar{\eta}_u^{u'}(k)\right)\right)-c\sum_{u\in N_u}\left(x_u(k)-\left(x_u(k)+\eta_u^{u'}(k)\right)\right)], \tag{8}$$

where μ_u is a constant step-size and $e_u(k+1):=b_u(k+1)-A_u^T(k+1)x_u(k)$ is a local priori error. $\eta_u^{u'}(k)$ and $\bar{\eta}_u^{u'}(k)$ denote the additive communication noise present in the reception of $x_{u'}(k)$ and $v_{u'}^{u}(k)$. **Algorithm 1** gives the description of the distributed least mean square algorithm. In detail, during the time instant $k+1$, node v_u receives the local estimates $\{x_{u'}(k)+\eta_u^{u'}(k)\}_{u'\in N_u}$ and plugs them into the equations above to evaluate $v_u^{u'}(k)$ for $u'\in N_u$. Each updated local Lagrange multiplier $\{v_u^{u'}(k)\}_{u'\in N_u}$ is subsequently transmitted to the corr-

▼**Table 2. Analysis of communication cost and time-to-completion**

Algorithm	Communication cost	Time-to-completion
D-MS	kmN^2	$km(N-1)$
D-MCGLS	$(k+1)(m+N)N+k(n+N)N$	$k(m+n+2)(N-1)$
D-CARP	$2knN$	$2n(N-1)+n\log N$
D-CE	knN	$knD_{\max}$
D-LMS	$kN(D_{avg}+1)$	$2kD_{\max}$
D-RLS	$(n+n^2)(N-1)$	$(n+n^2)(N-1)$

N is the network size, $m\times n(m\geqslant n)$ is dimensions of matrix A and k is the number of iterations (usually $m>>N$ and $n>>N$)
D-CARP: Distributed Component-Average Row Projection method.
D-CE: Distributed Cooperative Estimation methods.
D-LMS: Distributed Least Mean Squares method.
D-MCGLS: Distributed Modified Conjugate Gradient Least-Squares method.
D-MS: Distributed Multisplitting method.
D-RLS: Distributed Recursive Least-Squares method.

esponding neighbor $u' \in N_u$. Then upon reception of $\{v_{u'}^{u}(k) + \bar{\eta}_u^{u'}(k)\}_{u' \in N_u}$, the multipliers are jointly used along with $\{x_{u'}(k) + \eta_u^{u'}(k)\}_{u' \in N_u}$ and the newly acquired local data $\{b_u(k+1), A_u(k+1)\}$ to obtain $x_u(k+1)$ via the above equations. The $(k+1)$-st iteration is concluded after node v_u broadcasts $x_u(k+1)$ to its neighbors.

Algorithm 1: D-LMS Method

Each node v_u follows the same routines below
1. Arbitrarily initialize $\{x_u(0)\}_{i=1}^{N}$ and $\{v_u^{u'}(-1)\}_{i \in N}^{u' \in N_u}$
2. **While** not converged **do**
3. Broadcast $x_u(k)$ to neighbors in N_u
4. Update $\{v_u^{u'}(k)\}_{u' \in N_u}$
5. Transmit $v_u^{u'}(k)$ to each $u' \in N_u$
6. Update $x_u(k+1)$

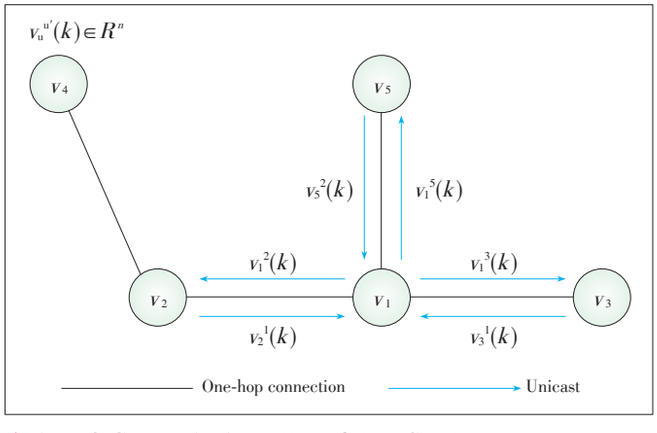
end while

1) Communication Cost

Algorithm 1 is simple and only steps 3 and 5 involve communication. Applied to system $Ax = b$, vectors $x_u(k)$ and $v_u^{u'}(k)$ are both of length n (columns of A). For example, in step 3, node v_1 needs to transmit $x_u(k)$ to all its neighbors; in step 5, v_1 needs to transmit different $v_u^{u'}(k)$ to different neighbors (**Fig. 2**). Suppose that the average degree of network is D_{avg}, in each iteration, the communication cost of one node is of $n(D_{avg}+1)$, so the communication of the network is $nN(D_{avg}+1)$. Suppose that after k iterations (the iteration number might be greater than the time instants, so after the k-th sample on node v_u is involved, the first sample is used as the $(k+1)$-st sample), the algorithm converges and the total communication cost is $knN(D_{avg}+1)$.

2) Time-to-Completion

In step 3 of Algorithm 1, node u needs to broadcast $x_u(k)$ to all its neighbors. From the receiver side, each node needs to receive different updates from all its neighbors, and then the delay of the whole network depends on the maximum node degree of the network since the algorithm is synchronous; this delay is $nD_{\max}$. In step 5, node u needs to send different $v_u^{u'}(k)$ to different neighbors, the delay is also $nD_{\max}$. The total communication delay is then $2knD_{\max}$.



▲**Figure 2. Communication pattern of D-LMS method.**

3.1.2 Distributed Recursive Least-Squares Method

Sayed and Lopes [14] developed a distributed least-squares estimation strategy by appealing to collaboration techniques that exploit the space-time structure of the data, achieving an exact recursive solution that is fully distributed. This D-RLS strategy is developed by appealing to collaboration techniques to achieve an exact recursive solution. It requires a cyclic path in the network to perform the computation node by node. The advantage of this method is the iteration number is fixed (the network size) for a give set of data to solve a least-squares problem, but the problem is a large dense matrix needs to be exchanged between nodes.

The details and analysis of D-RLS strategy are given in this section, and **Algorithm 2** gives the classic RLS procedure [15].

Algorithm 2: Recursive Least-Squares Procedure

Initial: $x^{-1} = \bar{x}$ and $P^{-1} = I$
1. **for** $k \geqslant 0$ **do**
2. $x^k = x^{k-1} + g^k\left[b(k) - A^T(k)x^{k-1}\right]$
3. $g^k = \dfrac{\lambda^{-1}P^{k-1}A(k)}{1 + \lambda^{-1}A^T(k)P^{k-1}A(k)}$
4. $P^k = \lambda^{-1}[P^{(k-1)} - g^k A^T(k)P^{(k-1)}]$

end for

To distribute the exact algorithm for estimating the vector x in the network of N nodes, each node v_u has access to regressors and measurement data $A_u(k)$ and $b_u(k), u = 1, ..., N$, where $b_u(k) \in \mathbb{R}$ and $A_u(k) \in \mathbb{R}^n$. At each time instant k, the network has access to space-time data:

$$b(k) = \begin{bmatrix} b_1(k) \\ b_2(k) \\ \vdots \\ b_N(k) \end{bmatrix} \text{ and } A(k) = \begin{bmatrix} A_1(k) \\ A_2(k) \\ \vdots \\ A_N(k) \end{bmatrix}, \tag{9}$$

where $b(k)$ and $A(k)$ are snapshot matrices revealing the network data status at time k. We collect all the data available up to time k into global matrices b and A:

$$b = \begin{bmatrix} b(0) \\ b(1) \\ \vdots \\ b(k) \end{bmatrix} \text{ and } A = \begin{bmatrix} A(1) \\ A(2) \\ \vdots \\ A(k) \end{bmatrix}. \tag{10}$$

Note that here is equivalent to solving the least-squares problem of $Ax=b$ by partition A and b row-wise and each node has one partition of consecutive rows of A and b. Applying the RLS algorithm is different to the D-LMS estimator since it gives the least-squares solution of the whole block of data $Ax=b$. Therefore, in the distributed RLS algorithm to solve a normal least-squares problem, we use $A_u(k)$ to indicate the row block on node v_u but not only one vector collected at time instant k (we can treat it as all the data collected till time k).

By assuming an incremental path is defined across the network cycling from node v_1 to v_2 and so forth, until node v_N, The RLS algorithm can be rewritten as a distributed version in **Algorithm 3** [14].

Algorithm 3: D-RLS Method

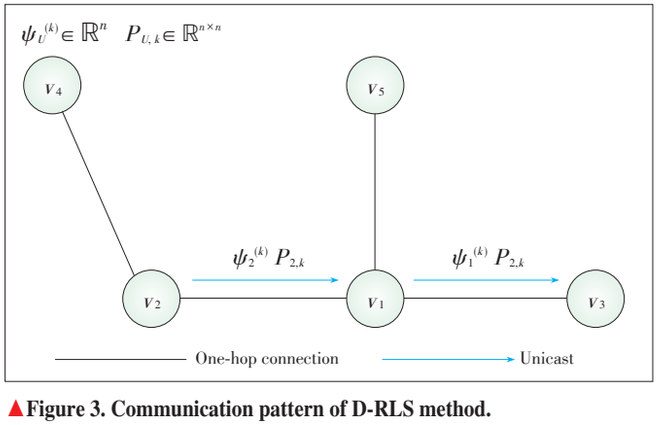
$\psi_0^{(k)}=x^{k-1}$, $P_{0,k}=\lambda^{-1}P^{k-1}$

1. **for** $u=1:N$, node v_u **do**
2. $e_u(k)=b_u(k)-A_u(k)\psi_{u-1}^{(k)}$
3. $\psi_u^{(k)}=\psi_{u-1}^{(k)}+\dfrac{P_{u-1,k}}{\gamma_u^{-1}+A_u^T(k)P_{u-1,k}A_u(k)}A_u(k)e_u(k)$
4. $P_{u,k}=P_{u-1,k}-\dfrac{P_{u-1,k}A_u(k)A_u^T(k)P_{u-1,k}}{\gamma_u^{-1}+A_u^T(k)P_{u-1,k}A_u(k)}$
5. If $u\neq N$ then
6. v_u send $\left\{\psi_u^{(k)},P_{u,k}\right\}$ to node v_{u+1}
7. **end if**

end for

1) Communication Cost

In the distributed RLS algorithm, the communication is in step 6. Each node shares with its successor node in the cycle path of network the quantities $\left\{\psi_u^{(k)},P_{u,k}\right\}$, where $\psi_u^{(k)}\in\mathbb{R}^n$ and $P_{u,k}\in\mathbb{R}^{n\times n}$. For example, node v_1 receives the message from v_2 and sends it to v_3 (**Fig. 3**). Therefore, in each iteration, the communication cost only happens in one node and it is $n+n^2$. Since the algorithm can converge after one cycle in the network, the total communication cost is $\left(n+n^2\right)(N-1)$. Note that a Hamiltonian path is required by this algorithm, and to find such a path, extra communication is required. This is another problem and out of the scope of the analysis in this paper, we omit this cost here.



▲**Figure 3. Communication pattern of D-RLS method.**

2) Time-to-Completion

In distributed RLS algorithm, it is easy to see that the delay in one step is $n+n^2$ and there are totally $N-1$ steps in the algorithm, so the total time-to-completion is $\left(n+n^2\right)(N-1)$.

3.2 Decentralized Optimization Methods

In recent years, much attention has been paid to fully distributed (decentralized) consensus optimization problems, especially in applications like distributed machine learning, multi-agent optimization, etc. Several algorithms have been proposed for solving general convex and (sub)differentiable functions. By setting the objective function as least-square, the decentralized least-square problem can be seen as a special case of the following model.

$$\min_{x\in\mathbb{R}^n}F(x):=\sum_{i=1}^{p}F_i(x), \tag{11}$$

where p nodes are in the network and they need to collaboratively estimate the model parameters x. Each node i locally holds the function F_i and can communicate only with its immediate neighbors.

Considering the problem in (11), (sub)gradient-based methods have been proposed [16]–[20]. However, it has been analyzed that the aforementioned methods can only converge to a neighborhood of an optimal solution in the case of fixed step size [21]. Modified algorithms have been developed in [16] and [17], which use diminishing step sizes in order to guarantee converging to an true solution. Other related algorithms were discussed in [22]–[28], which share similar ideas. The D-NC algorithm proposed in [16] was demonstrated to have an outer-loop convergence rate of $O\left(1/k^2\right)$ in terms of objective value error. The rate is same as the optimal centralized Nesterov's accelerated gradient method and decentralized algorithms usually have slower convergence rate than the centralized versions. However, the number of consensus iterations within outer-loop is growing significantly along the iteration. Shi [29] developed a method based on correction on mixing matrix for Decentralized Gradient Descent (DGD) method [21] without diminishing step sizes.

The algorithms mentioned above are based on synchronous models. Distributed optimization methods for asynchronous models have been designed in [30]–[32]. However, it is worth noting that their convergence rates are usually slower than the counterparts in synchronous models. In the next, we show the derivations of various decentralized optimization methods. In order to have a compact form, let $x\in\mathbb{R}^{np\times 1}:=\left[x_1^T,x_2^T,\cdots,x_p^T\right]^T$,

where x_i is a column vector containing local estimate of common interest x at node i. Similarly, define $F(x)=\sum_{i=1}^{p}F_i(x_i)$ and $\nabla F(x)\in\mathbb{R}^{np\times 1}:=\left[\nabla F_1(x_1)^T,\cdots,\nabla F_p(x_p)^T\right]^T$, where $\nabla F_i(\cdot)$ denotes the gradient of F_i.

Without loss of generality, we assume $n=1$ (the size of decision variable x), and the aforementioned problem can be formulated as:

$$\min_{x\in\mathbb{R}^p} F(x):Wx=x. \tag{12}$$

The constraint requires x to be consensual due to the property of the mixing matrix W. Now we first derive the DGD algorithm [21] from (5). Assuming W is symmetric and U is symmetric, we get $U^2=I-W$. Then $Wx=x$ if and only if $Ux=0$ holds. The original problem in (12) is thus equivalent to:

$$\min_{x\in\mathbb{R}^p} F(x):Ux=0. \tag{13}$$

If we use external penalty method, an unconstrained (but inaccurate) reformation can be expressed as:

$$\min_{x\in\mathbb{R}^p} F(x)+\frac{\rho}{2}\|Ux\|^2. \tag{14}$$

It is clear to see that (14) will approximate (15) as $\rho\to\infty$. Applying the gradient descent method to (14), we can obtain the following update rule:

$$\begin{aligned}x_{k+1}&=x_k-\alpha_k\left(\nabla F(x_k)+\rho_k(I-W)x_k\right)=\\&\left(1-\alpha_k\rho_k\right)x_k+\alpha_k\rho_k Wx_k-\nabla F(x_k).\end{aligned} \tag{15}$$

If choosing $\alpha_k\rho_k=1$ for all k, it yields the DGD algorithm:

$$x_{k+1}=Wx_k-\alpha_k\nabla F(x_k). \tag{16}$$

If we choose constant step-size $\alpha_k\equiv\alpha$, this is solving for the fixed $\rho=1/\alpha$, and hence not yielding the optimal consensus solution.

The DGD algorithm is shown in **Algorithm 4**.

Algorithm 4: DGD Method

Initialize $x_i^0,\forall i\in\{0,1,\cdots,p\}$.

1. **for** $k=0,1,\cdots$, node i **do**
2. $x_i^{k+1}=\sum_{j=1}^{p}W_{ij}x_j^k-\alpha\nabla F_i\left(x_i^k\right)$
3. Node i sends its updated value $x_i^{(k+1)}$ to its neighbors.

end for

The same situation happens in the algorithm D-NG (**Algorithm 5**) [16].

$$x_{k+1}=Wy_k-\alpha_k\nabla F(y_k), \tag{17}$$

$$y_{k+1}=x_{k+1}+\frac{k-1}{k+2}\left(x_{k+1}-x_k\right). \tag{18}$$

It is equivalent to applying (a special version of) Nesterov's gradient method [33] to the problem in (14) with increasing value of ρ (i.e. with diminishing α_k). Comparing (18) with (17), we can find that D-NG differs from DGD only in using y_k instead of x_k in (16). y_k defined in (11) is actually an extrapolation of the current x_k and the previous iteration x_{k-1}.

Algorithm 5: D-NG Method

Initialize $x_i^0,\forall i\in\{0,1,\cdots,p\}$.

1. **for** $k=0,1,\cdots$, node i **do**
2. $x_i^{k+1}=\sum_{j=1}^{p}W_{ij}x_j^k-\alpha\nabla F_i\left(x_i^k\right)$
3. Node i send its updated value x_i^{k+1} to its neighbors.

end for

We also show the derivation of Exact First-Order Algorithm (EXTRA) (**Algorithm 6**) [29]. We claim that EXTRA is equivalent to applying Alternating Direction Method of Multipliers (ADMM)method [34] to the original problem in (13). In (13), we would like to solve the exact constraint eventually. To this end, an unconstrained reformulation using augmented Lagrangian method can be described as:

$$\max_{z\in\mathbb{R}^p}\min_{x\in\mathbb{R}^p} F(x)-\rho z,Ux+\frac{\rho}{2}\|Ux\|^2. \tag{19}$$

Solving (19) by ADMM yields the following update equations:

$$z_{k+1}=z_k+Ux_k, \tag{20}$$

$$x_{k+1}=\arg\min_x\left\{F(x)+\frac{\rho}{2}\left\|Ux-z_k\right\|^2\right\}. \tag{21}$$

When a linearized preconditioned (approxiate $F(x)$ and linearize $\frac{\rho}{2}\|Ux-z_k\|^2$) version of x_{k+1} step is considered, (21) becomes:

$$\begin{aligned}x_{k+1}&=\arg\min_x\left\{\nabla F(x_k),x+\rho U(Ux_k-z_k),x+\frac{1}{2\alpha_k}\|x-x_k\|^2\right\}\\&=\left(I-\alpha_k\rho U^2\right)x_k+\alpha_k\rho Uz_k-\alpha_k\nabla F(x_k),\end{aligned} \tag{22}$$

where I denotes the identity matrix. Now for constant step-size $\alpha_k\equiv\alpha=1/\rho$, combining (20) and $U^2=I-W$ yields the EXTRA algorithm:

$$x_{k+1}=Wx_k-\alpha\nabla F(x_k)+\sum_{t=0}^{k-1}(I-W)x_t. \tag{23}$$

Note that (23) can also be seen as a "corrected" version of the DGD algorithm [29] comparing to (16).

Algorithm 6: EXTRA Method

Initialize $x_i^0, \forall i \in \{0,1,\cdots,p\}$. Set $\tilde{W} = (I+W)/2$.

1. **for** $k = 0,1,\cdots$, node i **do**
2. $x_i^{k+1} = \sum_{j=1}^{p} \tilde{W}_{ij} x_j^k - \alpha \nabla F_i(x_i^k)$
3. $x_i^{(k+2)} = \sum_{j=1}^{p} W_{ij} x_j^{(k+1)} - \alpha \nabla F_i(x_i^{(k+1)})$
4. Node i send its updated value x_i^{k+2} to its neighbors.

end for

Finally, we discuss the Fast Decentralized Gradient Descent (FDGD) method [35] that can be seen as an accelerated version of EXTRA. FDGD does not require diminishing step size and the method is accelerated to reach an optimal $O(1/k^2)$ convergence rate for general convex differentiable functions F_i. We adopte the idea of Nesterov's optimal gradient method for centralized smooth optimization [36] and mixing matrix method in network gossip and consensus averaging algorithms [29], [37]. In **Algorithm 7**, the superscript "ag" stands for "aggregated", and "md" stands for "middle". Matrix $\tilde{W} = (I+W)/2$ is a half-mixing matrix based on W. At iteration k, each node i sends its current x_k^i to all its immediate neighbors and receives x_k^j from them (one round of communication). The result $x^{i,ag}$ is output as the final solution. Algorithm 7 is a first-order method since only ∇F is required in each iteration, and hence the subproblem has low computation complexity. We do not need to use diminishing step sizes that converge to 0 but still can ensure both of convergence and consensus. Besides, if $\theta_k = 1$ for all k, Algorithm 7 reduces to a version very similar to regular decentralized gradient descent (16). However, by the choice of $\theta_k = O(1/k)$ as below, the change from input x_k^{md} to output x_{k+1}^{ag} is faster than that from x_k to x_{k+1}. This implies that Algorithm 7 can converge faster than regular DGD. The last remark explains intuitively why the multi-step scheme defined in Algorithm 7 can potentially accelerate the convergence. The comparison of various decentralized least-square methods is summarized in **Table 3**.

Algorithm 7: FDGD Method

Initialize $x_0^i, y_0^i = 0, x_0^{i,ag}, \forall i \in \{0,1,\cdots,p\}$. Set $\theta_k = \frac{2}{k+2}$, $L := \max\{L_i\}, \forall i$, where L_i is the Lipschitz constant of node i.

1. **for** $k = 0,1,\cdots$, node i **do**
2. $y_{k+1}^i = y_k^i + \sum_{j=1}^{p} (\tilde{W}_{ij} - W_{ij}) x_k^j$
3. $x_k^{i,md} = (1-\theta_k) x_k^{i,ag} + \theta_k \sum_{j=1}^{p} \tilde{W}_{ij} x_k^j$
4. $x_{k+1}^i = \sum_{j=1}^{p} \tilde{W}_{ij} x_k^j - y_{k+1}^i - \frac{1}{L\theta_k} \nabla F_i(x_k^{i,md})$
5. $x_{k+1}^{i,ag} = (1-\theta_k) x_k^{i,ag} + \theta_k x_{k+1}^i$
6. **end for**

Output $x_{k+1}^{i,ag}$

▼**Table 3. Communication cost and convergence speed comparison**

Algorithm	Communication cost	Convergence rate
DGD	$O(kN)$	Convergence to a neighborhood
D-NG	$O(kN)$	$O(\log k/k)$ for objective function
EXTRA	$O(kN)$	Ergodic rate $O(1/k)$ for residual
FDGD	$O(kN)$	$O(1/k^2)$ for objective function

N is the network size, and k is the number of communication.
DGD: Decentralized Gradient Descent EXTRA: Exact First-Order Algorithm
D-NG: Distributed Nesterov Gradient FDGD: Fast Decentralized Gradient Descent

4 Conclusions

In this paper, we surveyed some of the developments in distributed iterative methods and parallel iterative methods which can be potentially applied to solve least-squares problems in the mesh network. We covered the traditional iterative methods for solving linear systems including the relaxation methods, the conjugate gradient methods and the row action methods. One algorithm from each category is selected for describing how to apply them to solve least-squares problem in mesh network. We also surveyed some consensus and diffusion based strategies for parameter estimation in signal processing in the mesh network. Such the strategies only require local communications, however, for a large scale network, may take more iterations to converge to the required accuracy to reach an agreement among all the nodes. Algorithm selection depends on the context of the problem and the mesh network.

We also analyzed and compared the performance of the selected representative algorithms in terms of communication cost and time-to-completion. These two concerns are critical for evaluating the performance of distributed algorithms in the context of mesh networks. Besides, we think that a future research direction of distributed computing in mesh networks is the data loss tolerance: will the algorithm still approximate the optimal estimation x^* well if α-percent packets get lost in the network? Different from traditional parallel machines where data delivery is often guaranteed, in many distributed network applications, preventing data losses can either be very expensive (such as sensor networks) as it requires retransmissions, or have a time constraint in real-time applications (such as smart grid) that makes retransmitted data useless.

References

[1] A. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, USA: SIAM, 1996.
[2] L. Zhao, W. Z. Song, and X. Ye, "Decentralized consensus in distributed networks," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 20, pp. 1–20, 2016. doi: 10.1080/17445760.2016.1233552.
[3] L. Zhao, W. Z. Song, X. Ye, and Y. Gu, "Asynchronous broadcast-based decentralized learning in sensor networks," *International Journal of Parallel, Emer-*

gent and Distributed Systems, vol. 32, pp. 1–19, 2017.
[4] Y. Saad and H. A. van der Vorst, "Iterative solution of linear systems in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1–2, pp. 1–33, 2000. doi: 10.1016/S0377-0427(00)00412-X.
[5] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices of Numerical Mathematics and Scientific Computation*. Oxford, UK: Clarendon Press, 1989.
[6] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition. Philadelphia, USA: SIAM, 2003.
[7] C. C. Paige and M. A. Saunders, "LSQR: an algorithm for sparse linear equations and sparse least squares," *ACM Transactions on Mathematical Software*, vol. 8, no. 1, pp. 43–71, Mar. 1982.
[8] M. Baboulin, "Solving large dense linear least squares problems on parallel distributed computers. Application to the Earth's gravity field computation," Ph.D. Thesis, Toulouse Institute of Technology, Toulouse, France, 2006.
[9] M. T. Heath, E. Ng, and B. W. Peyton, "Parallel algorithms for sparse linear systems," *SIAM review*, vol. 33, no. 3, pp. 420–460, 1991.
[10] G. Mateos and G. B. Giannakis, "Distributed recursive least-squares: stability and performance analysis," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3740–3754, 2012.
[11] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Performance analysis of the consensus-based distributed LMS algorithm," *EURASIP Journal on Advances in Signal Processing*, 2009. doi: 10.1155/2009/981030.
[12] I. D. Schizas, G. B. Giannakis, S. I. Roumeliotis, and A. Ribeiro, "Consensus in ad hoc WSNs with noisy links—part II: distributed estimation and smoothing of random signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1650–1666, 2008. doi: 10.1109/TSP.2007.908943.
[13] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links—part I: distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008. doi: 10.1109/TSP.2007.906734.
[14] A. H. Sayed, and C. G. Lopes, "Distributed recursive least-squares strategies over adaptive networks. signals, systems and computers," *Fortieth Asilomar Conference on Signals, Systems and Computers (ACSSC' 06)*, Pacific Grove, USA, 2006, pp. 233–237. doi: 10.1109/ACSSC.2006.356622.
[15] V. K. Madisetti, *Digital Signal Processing Fundamentals*. Boca Raton, USA: CRC Press, 2009.
[16] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, May 2014. doi: 10.1109/TAC.2014.2298712.
[17] I.-A. Chen, "Fast distributed first-order methods," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, USA, 2012.
[18] I. Matei, and J. S. Baras, "Performance evaluation of the consensus-based distributed subgradient method under random communication topologies," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 754–771, 2011. doi: 10.1109/JSTSP.2011.2120593.
[19] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," in *IEEE 52nd Annual Conference on Decision and Control (CDC)*, Florence, Italy, 2013, pp. 6855–6860.
[20] A. Nedic and A. Olshevsky, "stochastic gradient-push for strongly convex functions on time-varying directed graphs," arXiv:1406.2075, 2014.
[21] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," arXiv:1310.7063, 2013.
[22] H. Terelius, U. Topcu, and R. Murray, "Decentralized multi-agent optimization via dual decomposition," in *18th IFAC World Congress*, Milano, Italy, 2011.
[23] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. Thesis, MIT, Cambridge, USA, 1984.
[24] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks. Information processing in sensor networks," in *Third International Symposium on Information Processing in Sensor Networks*, Berkeley, USA, 2004, pp. 20–27. doi: 10.1145/984622.984626.
[25] G. Shi and K. H. Johansson, "Finite-time and asymptotic convergence of distributed averaging and maximizing algorithms," arXiv:1205.1733, 2012.
[26] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.
[27] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. 4th International Symposium on Information Processing in Sensor Networks*, Piscataway, USA, 2005. doi: 10.1109/IPSN.2005.1440896.
[28] M. Zargham, A. Ribeiro, and A. Jadbabaie, "A distributed line search for network optimization," in *American Control Conference (ACC)*, Montreal, Canada, 2012, pages 472–477. doi: 10.1109/ACC.2012.6314986.
[29] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: an exact first-order algorithm for decentralized consensus optimization," arXiv:1404.6264, 2014.
[30] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem. "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," arXiv:1303.2837, 2013.
[31] E. Wei and A. Ozdaglar, "On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers," arXiv:1307.8254, 2013.
[32] L. Zhao, W.-Z. Song, L. Shi, and X. Ye, "Decentralised seismic tomography computing in cyber-physical sensor systems," *Cyber-Physical Systems*, vol. 1, no. 2–4, pp. 91–112, 2015. doi: 10.1080/23335777.2015.1062049.
[33] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course (Applied Optimization)*, 1st edition. New York, USA: Springer, 2004. doi: 10.1007/978-1-4419-8853-9.
[34] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2011. doi: 10.1561/2200000016.
[35] L. Zhao, W. Z. Song, and X. Ye, "Fast decentralized gradient descent method and applications to in-situ seismic tomography," *IEEE International Conference on Big Data*, Santa Clara, USA, 2015, pp. 908–917. doi: 10.1109/BigData.2015.7363839.
[36] Y. Nesterov, "Gradient methods for minimizing composite objective function," CORE Discussion Papers, 2007076, Catholic University of Louvain, Louvain-la-Neuve, Belgium, Sept. 2007.
[37] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006. doi: 10.1109/TIT.2006.874516.

Manuscript received: 2017-02-14

Biographies

SHI Lei (lshi1@student.gsu.edu) received his B.Sc. degree in software engineering from Tongji University, China in 2007, and the M.Sc. degree in computer science from Shanghai Jiao Tong University, China in 2010. He has received his Ph.D. degree from the Department of Computer Science of Georgia State University, USA. His research interests include wireless sensor networks, in-network processing and distributed systems.

ZHAO Liang (lzhao2@ggc.edu) received his Ph.D. in computer science from Georgia State University, USA and M.S in electrical engineering from Lehigh University, USA in 2016 and 2012, respectively. He is currently an assistant professor in computer science at Georgia Gwinnett College, USA. His current research interests are in the area of optimization and data analytics for distributed systems.

SONG Wenzhan (wsong@uga.edu) is now a professor at College of Engineering, University of Georgia, USA. His research mainly focuses on sensor web, smart grid and smart environment where sensing, computing, communication and control play a critical role and need a transformative study. His research has received 6 million+ research funding from NSF, NASA, USGS, Boeing, etc. since 2005. He is an IEEE senior member.

Goutham Kamath (gkamath1@student.gsu.edu) received his B.E. degree from India in 2009 and M.S. in electrical engineering from University of Wyoming, USA in 2012. He has received his Ph.D. degree in the Department of Computer Science of Georgia State University, USA. His research interests include wireless sensor networks, distributed systems and mobile ad-hoc networks.

WU Yuan (iewuy@zjut.edu.cn) received the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, China in 2010. He is currently an associate professor at the College of Information Engineering of Zhejiang University of Technology, China. His research interests focus on resource allocations for cognitive radio networks and smart grids.

LIU Xuefeng (csxfliu@comp.polyu.edu.hk) received the M.S. and Ph.D. degrees from Beijing Institute of Technology, China, and University of Bristol, United Kingdom, in 2003 and 2008, respectively. He is currently a research fellow at the Department of Computing of Hong Kong Polytechnic University, China. His research interests include wireless sensor networks, distributed computing, and in-network processing.