Finite-Time Distributed Averaging

Shaoshuai Mou A. S. Morse

Abstract—This paper proposes a distributed averaging algorithm for multi-agent networks, in which each agent is with a real-valued measurement. Provided that the underlying graph of the network is a tree, the proposed algorithm enables each agent to compute the average of the values of all agents in the network in a finite number of steps. Different from most existing finite-time distributed averaging algorithms, the algorithm proposed in this paper does not require each agent to know any global information.

I. INTRODUCTION

Considerable attention has been paid to consensus of multi-agent systems [1]–[6], in which each agent controls a scalar state initialized by a real-valued measurement and is able to communicate with certain other agents called its “neighbors”. One particular type of consensus algorithm is called the distributed averaging algorithm, which enable each agent to achieve the global average by only communications among neighbors [7]–[11]. By the global average is meant the average of initial state values of all agents across the network. Since measurements in practice are usually with zero-mean noise, a network of agents, which are equipped with distributed averaging algorithms and thus can compute the global average, usually constitute a much better sensor than a single agent.

One type of distributed averaging algorithms are gossiping algorithms [12], [13], in which pairs of nodes gossip at each iteration by updating their state values to the average of their previous ones. A more sophisticated approach to the problem is the Metropolitian algorithm [14], in which each node is usually assumed to be able to broadcast and update its state value to be a weighted average of states of all its neighbors at each iteration. Convergence of the Metropolis Algorithm depends on Metropolis weights, which require each agent to at least know an upper bound of degrees of all its neighbors. Metropolis algorithms usually converge faster than gossiping algorithms especially in fully connected networks, for which Metropolis algorithms converge in one step. But both of these two types of algorithms are usually with asymptotic convergence, that is, exact value of the global average can not be reached in a finite number of steps.

To further speed up the convergence and reduce the amount of information exchanged for the whole network to achieve the exact global average, we are interested in distributed averaging algorithms with finite time convergence in the framework of discrete time systems, which have recently gained more and more attention in the literature [15]–[21]. Such a finite-time distributed averaging algorithm was first briefly discussed in [15], in which the network needs to be fully connected at least one time-step. Later the authors of [16] have shown that each agent can calculate the global average as a linear combination of its own past values over a finite number of steps. To apply their method each agent either needs to know the minimal polynomial of an update matrix, which requires a strong knowledge of the underlying network topology, or at least performs a calculation of the minimal polynomial by each agent repeating over $n$ independent iterations, each for at least $n + 1$ time-steps, where $n$ is the number of agents in the network. The authors of [19] have obtained the convergence with a number of $O(n)$ time steps by employing “matrix factorization”, which is interesting but may not be easy to be decentralized. Recently the authors of [20] and [21] have proposed a distributed average algorithm based on graph Laplacian. The number of steps needed for convergence is equal to the number of distinct non-zero eigenvalues of the Laplacian. These eigenvalues can be computed distributively in [22] by carefully picking some parameters. But how to obtain the values of them in a finite number of steps without knowing the graph of the whole network is still unknown.

It’s the aim of this paper to propose a distributed algorithm which enables each agent to compute the exact value of the global average in a finite number of steps without requiring them to know any global information. The algorithm is inspired by the “ratio consensus” studied in [23], [24]. Here is the key idea: Let each agent $i$ be equipped with one real-value scalar state $x_i(t)$ and one integer scalar state $z_i(t)$; devise a local update for each agent such that all $x_i(t)$ and $z_i(t)$ converge to $\frac{\sum_{i=1}^{n} x_i(0)}{n}$ and $n$, respectively, in a finite number of steps; then the ratio $\frac{x_i(t)}{z_i(t)}$ converges to the global average. When the underlying graph of the network is a tree graph with diameter $d$, the proposed algorithm proves to reach the global average at most $d$ time steps. It’s also worthy mentioning that the proposed algorithm can be applied into solving the local average problem proposed in [25].

The remainder of this paper is organized as follows: in Section II, we formulate the distributed averaging problem and propose a finite-time distributed averaging algorithm. Then we present the main result in Section III. Comparison with other finite-time distributed averaging algorithms will be provided in Section IV and we conclude in Section V.

II. PROBLEM FORMULATION AND THE ALGORITHM

Consider a network of $n$-agents labeled by a set of integers $1, 2, \ldots, n$, in which each agent is able to exchange
information with certain other agents called its “neighbors”. Neighbor relations can be conveniently characterized by an undirected graph \( G \) with \( n \) vertices and a set of undirected edges defined so that there is an edge \((i, j)\) in \( G \) just in case that \( i \) and \( j \) are neighbors. \( G \) is called the underlying graph of the network.

Suppose each agent \( i \) is with a real-valued measurement \( x_{i0} \). In this paper we are interested in the following distributed averaging problem:

**Problem 1:** Devise an update rule for each agent to compute the global average

\[
y = \frac{1}{n} \sum_{j=1}^{n} x_{j0}
\]

in a finite number of time steps only by communication with its neighbors.

To solve Problem 1, we propose the following algorithm:

**A. A Finite-time Distributed Averaging Algorithm**

Let time be discrete in that \( t \) takes values from \( 0, 1, 2, \ldots \). Suppose each agent controls a real-valued state \( x_i(t) \in \mathbb{R} \) and a positive integer state \( z_i(t) \) with initialized \( x_i(0) = x_{i0} \) and \( z_i(0) = 1 \). Let \( \mathcal{N}_i \) denote the set of all the neighbors of agent \( i \). Let \( r_i \) denote the number of agent \( i \)'s neighbors, that is, \( r_i = |\mathcal{N}_i| \), where \(|\cdot|\) denotes the cardinality of a set. Between clock times \( t \) and \( t+1 \), each agent \( i \) takes actions following the steps enumerated below.

1. Agent \( i \) sends \( x_i(t) \) and \( z_i(t) \) to all its neighbors;
2. Agent \( i \) performs the following updates:

\[
x_i(t+1) = \begin{cases} 
x_i(0) + \sum_{j \in \mathcal{N}_i} x_j(0), & t = 0; \\
\sum_{j \in \mathcal{N}_i} x_j(t) + (1 - r_i)x_i(t-1), & t \geq 1.
\end{cases}
\]

and

\[
z_i(t+1) = \begin{cases} 
z_i(0) + \sum_{j \in \mathcal{N}_i} z_j(0), & t = 0; \\
\sum_{j \in \mathcal{N}_i} z_j(t) + (1 - r_i)z_i(t-1), & t \geq 1.
\end{cases}
\]

**Remark 1:** The additional integer state \( z_i(t) \) is employed for each agent \( i \) to obtain \( n \). If one assumes that each agent knows the number of agents in the network, the state \( z_i(t) \) and its update (2) can be omitted.

**III. MAIN RESULT**

In this section, we'll give the main result of this paper. Toward to this end, we need some definitions in graph theory. In an undirected graph, the union of edges \((i, i_1), (i, i_2), \ldots, (i, j)\) is called a path of length \( k \) between \( i \) and \( j \). An undirected graph is a tree if there is only one path between any pair of vertices. In a tree graph, the distance between two vertices \( i \) and \( j \) denoted by \( d_{ij} \) is defined as the length of the path between them. In an undirected graph, one has \( d_{ij} = d_{ji} \) and specially \( d_{ii} = 0 \). The diameter of a tree graph is

\[
d = \max_{i=1,2,\ldots,n} d_i \quad (3)
\]

where

\[
d_i = \max_{j=1,2,\ldots,n} d_{ij}
\]

Let \( \mathcal{N}_{ik} \) be the set of all vertices such that \( d_{ij} \leq k \) for each \( j \in \mathcal{N}_{ik} \). Obviously, \( \mathcal{N}_{i0} = \{i\}, \mathcal{N}_{i1} = i \cup \mathcal{N}_i \), and

\[
\mathcal{N}_{ik} = \{1, 2, \ldots, n\}, \quad k \geq d_i
\]

The following is our main result:

**Theorem 1:** Suppose \( G \) is a tree graph with the diameter equal to \( d \). By the proposed algorithm, one has

\[
x_i(t) = \frac{1}{|\mathcal{N}_{it}|} \sum_{j \in \mathcal{N}_{it}} x_{j0}, \quad t \geq 0 \quad \text{(4)}
\]

From \( \mathcal{N}_{it} = \{1, 2, \ldots, n\} \) for \( t \geq d_i \) and (3), one has

\[
x_i(t) = \frac{1}{n} \sum_{j=1}^{n} x_{j0}, \quad t \geq d \quad \text{(5)}
\]

which together with (4) implies that

\[
x_i(t) = \frac{1}{n} \sum_{j=1}^{n} x_{j0}, \quad t \geq d \quad \text{(6)}
\]

Therefore the proposed algorithm enables each agent to obtain the exact value of the global average at most \( d \) steps. Once the agent reaches the global average, it will hold it. Thus the proposed finite-time distributed averaging algorithm solves Problem 1.

To prove Theorem 1 is true, we let \( A = [a_{ij}]_{n \times n} \) be the adjacency matrix of \( G \), that is, \( a_{ij} = a_{ji} = 1 \) if there is an undirected edge \((i, j)\) in \( G \); otherwise, \( a_{ij} = 0 \). Let \( x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]' \) and \( z(t) = [z_1(t), z_2(t), \ldots, z_n(t)]' \). From (1) and (2), one obtains the \( x \)-system

\[
x(1) = (I + A)x(0)
\]
\[
x(t+1) = Ax(t) + (I - D)x(t-1), \quad t \geq 1
\]

and the \( z \)-system

\[
z(1) = (I + A)z(0)
\]
\[
z(t+1) = Az(t) + (I - D)z(t-1), \quad t \geq 1
\]

where \( I \) is the \( n \times n \) identity matrix and \( D = \text{diag} \{r_1, r_2, \ldots, r_n\} \).

To further simplify the \( x \)-system and the \( z \)-system. We define a sequence of matrices \( S(t) \), \( t = 0, 1, 2, \ldots \), such that

\[
S(0) = I, \quad \text{(7)}
\]
\[
S(1) = I + A, \quad \text{(8)}
\]
\[
S(t) = AS(t-1) + (I - D)S(t-2), \quad t \geq 2 \quad \text{(9)}
\]

Then one has

\[
x(t) = S(t)x(0) \quad \text{(10)}
\]
\[
z(t) = S(t)z(0) \quad \text{(11)}
\]

More can be said about \( S(t) \) in the following lemma.
Lemma 1: Suppose G is a tree graph. Let $S(t) = [s_{ij}(t)]_{n \times n}$ for $t = 0, 1, 2, \ldots$. Then

$$s_{ij}(t) = \begin{cases} 1, & d_{ij} \leq t; \\ 0, & d_{ij} \geq t + 1. \end{cases}$$ (12)

Proof of Lemma 1: We prove this lemma by induction. Since $S(0) = I$ and $S(1) = I + A$, one has (12) holds for $t = 0$ and $t = 1$.

Suppose (12) holds for $t \leq k$ with $k \geq 1$. By (9) one has for $t = k + 1$,

$$s_{ij}(k + 1) = \sum_{v \in N_i} s_{uj}(k) + (1 - r_i)s_{ij}(k - 1)$$ (13)

If $d_{ij} \leq t$, one only needs to consider the following three cases since $t = k + 1$:

- Case 1: $d_{ij} = k + 1$. Note that there is only one path between $i$ and $j$. Then there is only one vertex $u \in N_i$ such that $d_{uj} = k$ and for all $v \in N_i, v \neq u$, one has $d_{vj} = k + 2$. Thus $s_{uj}(k) = 1$ and $s_{uj}(k) = 0$ for $v \in N_i, v \neq u$. It follows that

$$\sum_{v \in N_i} s_{uj}(k) = s_{uj}(k) = 1$$

which together with $s_{ij}(k - 1) = 0$ by the induction assumption and (13) implies that $s_{ij}(k + 1) = 1$ in this case;

- Case 2: $d_{ij} = k$. Since there is only one path between $i$ and $j$, there is only one vertex $u \in N_i$ such that $d_{uj} = k - 1$ and $d_{vj} = k + 1$ for all $v \in N_i, v \neq u$. Thus $s_{uj}(k) = 1$ and $s_{uj}(k) = 0$ for $v \in N_i, v \neq u$. It follows that

$$\sum_{v \in N_i} s_{uj}(k) = s_{uj}(k) = 1$$

by which and $s_{ij}(k) = 0$ by the induction assumption and (13), one has $s_{ij}(k + 1) = 1$ in this case;

- Case 3: $d_{ij} \leq k - 1$. One has $d_{ij} \leq k$ for all $v \in N_i$. By the induction assumption, one has $s_{uj}(k) = 1$ for all $v \in N_i$ and $s_{ij}(k - 1) = 1$. Then

$$\sum_{v \in N_i} s_{uj}(k) = r_i$$

It follows from (13) that

$$s_{ij}(k + 1) = r_i + (1 - r_i) = 1$$

To sum up the above three cases, one has $s_{ij}(k + 1) = 1$ for $d_{ij} \leq t$.

In the other hand, if $d_{ij} \geq t + 1$, that is, $d_{ij} \geq k + 2$, one has $s_{ij}(k - 1) = 0$ and $s_{uj}(k) = 0$ for all $v \in N_i$. Then by (13) one has $s_{ij}(k + 1) = 0$. We complete the proof. 

Proof of Theorem 1: From the definition of $N_{it}$, one has $d_{ij} \leq t$ if and only if $j \in N_{it}$. Then (12) can be rewritten as

$$s_{ij}(t) = \begin{cases} 1, & j \in N_{it}; \\ 0, & j \notin N_{it}. \end{cases}$$ (14)

which together with (10) and (11) leads to

$$x_i(t) = \sum_{j \in N_{it}} x_j(0)$$

$$z_i(t) = \sum_{j \in N_{it}} z_j(0) = |N_{it}|$$

Then (4) is true and we complete the proof.

IV. EVALUATION OF THE ALGORITHM

In this section, we’ll evaluate the proposed algorithm from the following three aspects:

A. Comparison with Other Finite-time Algorithms

We first compare the proposed algorithm with the flooding algorithm mentioned in [14]. By Theorem 1, the algorithm proposed in this paper solves the distributed averaging problem in $d$ time-steps, where $d$ denotes the diameter of the tree graph. This is as fast as the flooding algorithm, in which each agent maintains a set of real values and a set of agent labels and at each time step each agent exchanges these two sets with its neighbors. The flooding algorithm requires each agent to be able to storage at least $n$ real numbers and $n$ agent labels. When the number of agents in the network is large, this leads to a requirement for large storage and a huge amount of information exchanged. Compared with this, in the proposed algorithm each agent only needs to store two real numbers $x_i(t), x_i(t - 1)$ and two positive integers $z_i(t), z_i(t - 1)$. Moreover, at one time step each agent sends only one real number and one positive integer to its neighbors. For a network consisting of $n \geq 3$ agents, the proposed algorithm outperforms the flooding algorithm in both storage requirement and the amount of information exchanged.

Second, we provides comparisons with results in [16] and [20]. For a given connected graph $G$, the algorithm proposed in [20] needs $p(L)$ steps to obtains the global average, where $p(L) + 1$ denotes the number of distinct eigenvalues of the Laplacian matrix $L$ of $G$. Note that the relation between the number of distinct eigenvalues of the graph Laplacian and its diameter $d$ is $d + 1 \leq p(L)$, that is,

$$d \leq p(L)$$ (15)

Then the algorithm proposed in this paper is faster than the one in [20]. It is also worthy mentioning that the proposed algorithm does not require each agent to known any global information such as the minimal polynomial of the update matrix in [16] or the Laplacian of the graph in [20].

B. Application into a Local Average Problem

In a recent paper [25], the authors have proposed algorithms to solve the following Distributed Local Averaging Problem exponentially fast for 1D networks:

**Problem 2:** For a given positive integer $l$, devise an update rule for each agent $i$ to compute the local average

$$y_i(l) = \frac{1}{|N_{it}|} \sum_{j \in N_{it}} x_j(0)$$ (16)
In problem 2, each agent $i$ is only interested in the average of values from agents in $N_{l}$, which are agents $l$ distance away from $i$, and all the values farther than $l$ are ignored. Problem 2 is relevant in practice in the sense that the global average may suppress too much information of interest. For example, one thousand agents constitute a network to monitor the air-pollution across a city. Besides giving a single reading on the global average of the air pollution across the whole city, one might want to identify hotspots of pollution of the city, in which case the form of a local average might be useful. Theorem 1 shows that the algorithm in this note solves Problem 2 in 2D in a finite number of time steps when $G$ is a tree graph.

C. When the underlying graph is connected but not a tree

We note that the proof of Theorem 1 only holds when $G$ is a tree graph. When $G$ is a connected graph with cycles, one needs to find a spanning tree of $G$ before applying the proposed algorithm. The same requirement was made in [19]. The problem of finding a minimum spanning tree in a connected graph has been well studied in the field of computer science [27], [28]. The most classic distributed algorithm for this was proposed by Gallager et al in [29], which solves the problem in $O(n \log n)$ time. Later the solution was improved to $O(n)$ in [30].

V. Conclusion

Based on the idea of “ratio consensus”, we have proposed a finite-time distributed averaging algorithm which enables all agents to compute the global average in a finite number of steps under the condition that the network is a tree.

In future we will consider applying the idea “ratio consensus” into improving results of [31]–[33], which may lead to distributed algorithms of solving linear equations in a finite number of steps.

VI. Acknowledgements

The authors are grateful to Professor Daniel A. Spielman for helpful discussions on algorithms of finding a spanning tree.

References