2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Consensus-based Data Statistics in Distributed Network Systems

Yifan Cai†, Jianping He‡, Wenbin Yu‡ and Xinping Guan‡

Abstract— **Data has become increasingly important in network systems because a lot of data is needed in new technologies such as machine learning. To obtain statistics (e.g. maximum, average, and distribution) in a fully distributed way with low complexity is challenging. Existing research on consensus algorithms can successfully obtain the max/min, average and median in a distributed network, but little work has been done on how to compute other statistics, especially probability density function (PDF). In this paper, consensus-based algorithms are proposed to obtain PDF in a fully distributed way with low complexity. The key idea of our algorithms is to divide the range of nodes' values into several sections and calculate the proportions of values in each section in a fully distributed way. If nodes have their unique identifications (IDs), repeatedly run max/min consensus algorithm in the network to reach the partially max/min values and then erase them in order to reach all values exactly once. We prove that the algorithm converges in finite time. When nodes' IDs are not available, the main challenge is to solve the conflicts when two or more nodes have the same value. We propose an asymptotically converged algorithm to solve the problem in this scenario.**

I. Introduction

With the development of computing technology, a growing number of enterprises and institutions have begun to collect and exploit data in network systems [1]. Data has become increasingly important in various areas, under which conditions, detailed and effective statistics such as the maximum and average values are expected to be obtained. These statistics provide an essential knowledge of data. For instance, sensors deployed in different areas of a lake collect the local state of water quality and cooperatively compute the statistic information, e.g., the average, variance, to show the global water quality of the lake [2]. Also in the field of machine learning, an essential task is to collect a large number of parameters to train models [3]. However, it is still far from having an insight of the data. The maximum, minimum, and average reveal only a little information. In order to make full use of data, more statistics of the data should be considered.

The distribution provides the property of data. After the distribution is obtained, not only are we able to get most of the statistics, but also to do much further analysis of the data (for example, after computing the proportions of each age group, more proper advertisements can be provided on a certain website [4]). To obtain the distribution of data, there is a straightforward approach in centralized network systems. That is, the central node simply collects all values in the network and calculates the distribution. However, in distributed networks, no nodes have access to global information, making the problem more difficult. It is well known that distributed networks are more promising systems and have lots of advantages over centralized ones. Firstly, distributed networks have better robustness, e.g., the network will not be strongly affected when there is an attack or power failure. Secondly, the bandwidth of each node, which is based on the number of neighbors of nodes, will be much smaller in distributed networks than that in centralized ones. In addition, distributed networks are more scalable, nodes can easily join or quit from the network without affecting the whole construction a lot. Considering these advantages and the importance of data statistics, it is worthwhile to investigate how to obtain the different statistics of data in a fully distributed way.

To calculate statistics, consensus algorithms are widely used in distributed network systems. Plenty of research has been conducted so far on how to compute statistics such as the maximum and minimum [5], [6], the average [7]–[11], the median [12], [13] in static networks. These algorithms also work under the considerations of dynamic topologies of networks [14]–[17], privacy concerns [5] and so on. Nevertheless, little research has been conducted on calculating variance and distribution. Computing distribution is especially difficult. Firstly, it cannot be obtained by simply calculating some parameters (e.g., μ and σ in Gaussian distribution) because the type of distribution is not known beforehand. Secondly, much storage will be consumed if the protocol is simply designed to collect all values in one node. Lastly, no global information is known to each node, so the distribution needs to be obtained in a completely distributed way. Recently, a distributed solution for calculating the distribution has been worked out in [18] to estimate the distribution using stochastic approximation. This method works when the data range is a finite set and known to all nodes beforehand. It also records distribution for every unique value to ensure that the algorithm is accurate. However, in real situations, nodes do not know the range of values beforehand. In addition, this method costs a lot of storage space to store the distribution of every possible value, where the number of possible values can be close to infinity in real-life networks.

To solve these existing challenges, this paper aims to design distributed algorithms to obtain the distribution of data. To calculate the distribution, we find that the range of data values, i.e. the difference between maximum and

†: School of Software Engineering, Shanghai Jiao Tong University, Shanghai, China fyc1007261@sjtu.edu.cn

‡: The Dept. of Automation, Shanghai Jiao Tong University, and the Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai, China jphe@sjtu.edu.cn, yuwenbin@sjtu.edu.cn, xpguan@sjtu.edu.cn

This work was supported in part by the Natural Science Foundation of China (NSFC) under grant 61773257, 61761136012, and 61803261.

minimum, can be obtained from using max/min consensus. Then, we can divide the range into a number of sections, and then calculate the number of data values in each section to obtain the distribution approximately. Based on this observation, we design consensus-based algorithms to obtain the data distribution by considering different network scenarios. In summary, the main contributions and approaches of this paper are listed as follows.

- We proposed efficient consensus-based algorithms for each node to calculate the variance, median, and PDF of data in network systems. To best of our knowledge, it is the first time to obtain the PDF statistic in a fully distributed way within finite time.
- We proved the convergence of proposed algorithms, and analyzed the storage cost and accuracy of them. It is shown that the algorithms have low computation complexity and high accuracy.
- Extensive simulations were conducted to verify the correctness of the theoretical results and show the efficiency of the proposed algorithms.

This paper is organized as follows: models, existing consensus algorithms and problem formulation are reported in Section II. Section III provides the main results, including PDF calculating algorithms and the analysis of them. The simulation results are given in Section IV, and the whole paper is summarized in Section V.

II. Preliminaries And Problem Formulation

A. Network Model

A distributed network system is modeled by an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Denote n as the number of nodes in the network, and d as the diameter. Denote d_i as the degree of node i and N_i as the neighbor set of node i, where $j \in N_i$ if and only if $(i, j) \in E$. Assume that all operations in the network are synchronous and the network is connected with a fixed topology.

Let $x_i(k)$ be the state of node i in iteration k and $\mathbf{x}(k)$ as the vector of values. Let $x_i(0)$ be the initial state of node i. In iteration k, each node in the network broadcasts its current state $x_i(k)$, and updates its state iteratively following a designated protocol.

B. Classic Consensus Algorithms

There are mainly two types of consensus algorithms, which are max/min consensus and average consensus algorithms.

Mathematically, a maximum consensus algorithm is described as follows,

$$x_i(k+1) = \max\{x_i(k), x_j(k) | i \in V, j \in N_i\}$$

i.e., in each iteration, each node updates its value by the maximum of itself and its neighbors. It is proved that maximum consensus is achieved after d iterations when the network is interconnected with a fixed topology [20].

For average consensus, at each iteration, node i updates its value as follows,

$$x_i(k+1) = W_{ii}x_i(k) + \sum_{j \in N_i} W_{ij}x_j(k), i \in V$$

It can be rewritten in the matrix form, which is given by $\mathbf{x}(k+1) = W\mathbf{x}(k)$, where $W \geq 0$ is a non-negative weight matrix, satisfying $W_{ij} = 0$ for $j \notin N_i$, $W_{ii} = 1 - \sum_{j \in N_i} W_{ij}$, $W\mathbf{1} = 1$, and $\mathbf{1}W = 1$, where $\mathbf{1}$ denotes the vector of all ones. Clearly, W is a double stochastic matrix.

An often used weight matrix guaranteeing asymptotic average convergence is *Metropolis weight* (see [15], [19]),

$$W_{ij} = \begin{cases} \frac{1}{1+\max\{d_i, d_j\}}, & \{i, j\} \in E, i \neq j; \\ 1 - \frac{1}{N}\sum d_i, & i = j; \\ 0, & otherwise, \end{cases}$$

in which case node i only needs to know its degree d_i to setup the related values. Therefore, it is widely used in distributed network systems.

It has been proved in [19] that with Metropolis weight, the average consensus is achieved asymptotically, i.e.,

$$\lim_{k \to \infty} x_i(k) = \bar{x} \tag{1}$$

C. Problem Formulation

Classic consensus algorithms solve the problem of calculating the max/min and average of values in a distributed network system. However, the existing consensus algorithm cannot be used to obtained the variance and distribution directly. Thus, this paper aims to solve these problems of computing variance and distribution by exploiting the idea of consensus algorithms.

Considering the variance first, denoted by σ^2, it has the following property that

$$\sigma^2 = E(x^2) - E^2(x) = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \bar{x}^2$$

where $E(x)$ represents the mathematical expectation of x. Therefore, the problem of calculating the variance is transformed into calculating the average of x and x^2, which can be calculated easily with average consensus algorithms.

As for the PDF, since the type of distribution that the values follows is unknown, i.e. we do not know whether the values follow normal distribution, uniform distribution, or other ones. As a result, we cannot estimate some parameters to get the PDF and it is hard to directly get the PDF with closed-form of expression. As the data in the network is in a discrete form, we aim to obtain the discrete distribution of data in the paper.

In the following sections, we focus on the PDF estimation, and design consensus-based algorithms to solve the challenges in distributed ways.

III. Main Results

Network systems can be classified into two categories, depending on whether their nodes have IDs. In some networks, IDs are unique values which distinguish each node from others, for example, the MAC addresses in computer networks, or the device IDs in sensor networks. There are also networks without IDs, e.g., in social networks, anonymous users are unwilling to share their IDs. Considering the difference, we propose two different methods to calculate the PDF.

A. PDF Calculating Algorithm with IDs

In this case, the calculation of the PDF includes four key procedures: i) using the max/min consensus to calculate the nodes' data values range, i.e., the difference between the maximum and minimum; ii) dividing the range of data values into several sections; iii) repeatedly running the max/min consensus to calculate the maximum and minimum values among existing uncounted data and calculate the proportions of values in corresponding sections; and iv) excluding the maximum and minimum values from the existing uncounted data set with the help of ID marking technology, which ensures that all nodes' values are used only once.

Specifically, we divide the range of data values, denoted by r, evenly into s (how to choose it will be discussed in Sec. III-C) sections. Denote $\rho[i,j](k)$ as the temporary estimate in node i of proportion of nodes' values in the j-th section in iteration k, and denote $I_i^{\max}, I_i^{\min}$ as the ID of max/min stored in node i. In each cycle, nodes broadcast their values and IDs, in order to cooperatively calculate the maximum and minimum using the maximum/minimum consensus algorithm and store the sources' IDs. Then, we introduce ID marking technology. When several values are equal, compare the ID and store the largest(in max consensus protocol)/smallest(in min consensus protocol) ones. For example, nodes A and B have the same value 100, while the IDs of nodes A and B are 18 and 17, respectively. Then in the max consensus procedure, nodes store the ID of node A and regard node A as the source of maximum. Set $x_i^{max}(t,0)$ and $x_i^{min}(t,0)$ to be the initial value of node i for each cycle t. The update rule is shown as follows.

$$x_i^{\max}(t,k+1) = \max\{x_i^{\max}(t,k), x_j^{\max}(t,k)\}, j \in N_i \quad (2)$$

and for minimum, we have

$$x_i^{\min}(t,k+1) = \min\{x_i^{\min}(t,k), x_j^{\min}(t,k)\}, j \in N_i \quad (3)$$

Sources of the max and min values erase their values at the end of each cycle, and the one or two sections which contain the two values are selected and the proportions of values in the sections are modified.

Using the properties of PDF, we have

$$\int_\alpha^\beta f(x)dx = \Pr(\alpha < x \le \beta) \quad (4)$$

Let α, β be the left and right borders of section j. Then,

$$\rho[i,j] = \Pr(\alpha < x \le \beta) \quad (5)$$

Because the entire range r is divided into s sections,

$$\beta - \alpha = \frac{r}{s} \quad (6)$$

As $\rho[i,j](k)$ is constant in each section, from (6), (7) and (8), the proportion and the PDF follow the relationship that $\int_\alpha^\beta f(x)dx = \frac{r}{s}f(x) = \rho[i,j]$ Therefore, the value of PDF y of node i in section j can be computed by $y_i = f(x) = \frac{s}{r}\rho[i]$ The details of the above whole procedure is described in Algorithm 1 as follows.

Algorithm 1 : *PDF Calculating Algorithm with IDs*

Run max/min consensus to get the global $x_{\max}$ and $x_{\min}$
$r = x_{\max} - x_{\min}$
for $i \in V$ **do**
  $x_i^{\max}(0), x_i^{\min}(0) \leftarrow x_i(0)$,
  $I_i^{\max}(0), I_i^{\min}(0) \leftarrow i$
end for
for $i \in V$ and $j \leftarrow 1$ to s **do**
  $\rho[i,j] \leftarrow 0$
end for
for $t \leftarrow 1$ to $\lfloor \frac{n+1}{2} \rfloor$ **do**
  for $i \in V$ **do**
    Input $x_i^{\max}(t)$, $x_i^{\min}(t)$ from max/min algorithms with IDs.
    Sources of max and min erase their values.
    if $t = \frac{n+1}{2}$ and n is odd **then**
      $\rho[i, \lfloor (x_i^{\max}(t) - x_{\min})\frac{s}{r} \rfloor + 1] += \frac{1}{n}$
    else
      $\rho[i, \lfloor (x_i^{\max}(t) - x_{\min}\frac{s}{r} \rfloor + 1] += \frac{1}{n}$
      $\rho[i, \lfloor (x_i^{\min}(t) - x_{\min})\frac{s}{r} \rfloor + 1] += \frac{1}{n}$
    end if
  end for
end for
Output $\frac{s}{r}\rho[i]$

Next, we prove the finite time convergence of Algorithm 1 and obtain a theorem as follows.

Theorem 3.1: The PDF is obtained distributedly by Algorithm 1 in $(n-1)\lfloor \frac{n+1}{2} \rfloor$ iterations.

Proof: Let $\boldsymbol{a}$ be a vector of size n which satisfies

$$\forall i \in [1,n), a_i \le a_{i+1}; \forall a_i = a_{i+1}, I_i \le I_{i+1}$$

In each cycle, nodes in the network run the max/min consensus algorithm to get the partially max/min values. After that, sources of the max/min erase their values. Therefore, in cycle t, a total of $(t-1)$ values in the front of the vector and $(t-1)$ values in the end have been obtained and erased in previous cycles. As a result, nodes obtain the t-th maximum and t-th minimum values in cycle t, which is described as

$$\begin{cases} x_i^{\min}(t) = a_t \\ x_i^{\max}(t) = a_{n-t+1} \end{cases} \quad (7)$$

where $(n-1)$ iterations are needed for each cycle to achieve the max/min consensus excluding the erased ones.

As the range of each section is $\frac{r}{s}$, it is not difficult to calculate that any value obtained, denoted by x, is in section $(x - x_{min})\frac{s}{r} + 1$. The one or two sections which $x_i^{\min}(t)$ and $x_i^{\max}(t)$ are in then have a proportion that is $\frac{2}{n}$ (when two values are in the same section) or $\frac{1}{n}$ (in different sections)

larger. There is a special case when $t = \frac{n+1}{2}$ and n is odd, which means that $x_i^{\min}(t)$ and $x_i^{\max}(t)$ are exactly the same value from the same node. Steps 14 and 15 ensure that the proportion is modified only once in this special case.

After $\lfloor \frac{n+1}{2} \rfloor$ cycles, all values have been obtained, where $(n-1)$ iterations are needed for max/min consensus in each cycle. Therefore, all values are obtained after a total of $(n-1)\lfloor \frac{n+1}{2} \rfloor$ iterations. The proportions and then the PDF are calculated, which has completed the proof. ■

It is shown in Theorem 3.1 that the PDF is obtained by each node in $(n-1)\lfloor \frac{n+1}{2} \rfloor$ iterations with Algorithm 1. The algorithm converges in a finite time and the convergence time is only affected by the size of network n. From (4) and (5), the partial maximum and minimum are obtained in each cycle, and they eventually reach the middle of the sequence when the algorithm is stopping. Therefore, with the idea in Algorithm 1, the median is obtain in nature. Denote M as the median of the value, we thus have a theorem as follows, which can be achieved using the results in [20]

Theorem 3.2: The median value is obtained in $(n-1)\lfloor \frac{n+1}{2} \rfloor$ iterations by each node with (4) and (5), and thus

$$M = \frac{1}{2}(x_i^{\max}(\lfloor \frac{n+1}{2} \rfloor, n-1) + x_i^{\min}(\lfloor \frac{n+1}{2} \rfloor, n-1))$$

holds for $\forall i \in V$.

If the diameter of the network d is known to all nodes beforehand, as the max/min consensus algorithm converges in d iterations, only d rather than $(n-1)$ iterations are required in step 12. Then, the PDF (and median) is obtained by Algorithm 1 in $\lfloor \frac{n+1}{2} \rfloor d$ iterations. Compared to existing work of median calculating (e.g. an algorithm with time complexity $O(nr)$ proposed in [12]), our method has the advantage that it converges in a finite time (not affected by the range of values, etc.) as long as the topology is fixed.

B. Generic PDF Calculating Algorithm

When IDs of nodes are not available, it is hard to solve the conflicts on equal values in order to make sure that every value is obtained and used only once. Therefore, in this algorithm, nodes do not try to obtain every value, instead they broadcast their temporary estimate of the distribution and update it in each iteration. We initialize value $\rho[i,j](0)$ to 1 if the value of node i is in section j, otherwise it is set to be 0. The average value of $\rho[i,j]$ for $i \in V$ is the proportion of values which are in section j. The proportion matrix ρ in iteration k is estimated by $\rho(k+1) = W\rho(k)$, where W is a Metropolis weight matrix. From (1), it follows that ρ asymptotically converges, i.e.

$$\lim_{k\to\infty} \rho[i](k) = \frac{1}{n}\sum_{i=1}^{n} \rho[i](0), i \in V$$

which is the initial proportions of values in all sections. In the same way, we have $y_i = f(x) = \frac{s}{r}\rho[i]$, which is the value of PDF.

The procedure is described in Algorithm 2 as follows.

Both methods in [18] and Algorithm 2 are based on average consensus algorithm to update the values. However,

Algorithm 2 : *Generic PDF Calculating Algorithm*

1: Input: $x_i(0)$, num_iter, W.
2: Input $x_{\max}, x_{\min}$ obtained from max/min consensus algorithms.
3: $r = x_{\max} - x_{\min}$
4: **for** $i \in V$ **do**
5: **for** $j \leftarrow 1$ to s **do**
6: $\rho[i,j](0) \leftarrow 0$
7: **end for**
8: $\rho[i, \lfloor (x_i(0) - x_{\min})\frac{s}{r} \rfloor + 1](0) \leftarrow 1$
9: **end for**
10: **for** $k \leftarrow 0$ to num_iter **do**
11: $\rho(k+1) = W\rho(k)$
12: **end for**
13: Output $\frac{s}{r}\rho[i]$.

Algorithm 2 successfully solves the problem of the ignorance of global value range. There is also a way to balance the storage cost and accuracy for this algorithm, which will be discussed in detail in the next subsection.

C. Cost of Storage and Accuracy

In both algorithms, we calculate the proportions of data in the sections to estimate the PDF. Therefore, the value of PDF in each section, which is a function of proportion in it, is a constant value. However, the actual distribution in the section can be in various forms. Thus, it is not an absolutely accurate algorithm.

Denote δ as the precision of the data. For example, integer values have a δ of 1. Denote $\bar{y}$ as the value of PDF in a specific section j. For convenience, let $m = \frac{r}{s\delta}$, so m is the maximum number of sub-sections that can be divided in a section with a range of $\frac{r}{s}$. In order to reach a higher accuracy, we divide section j further into α (a positive integer) sub-sections. Define $Error_j(\alpha) = \frac{1}{m}\sum_{i=1}^{m}(\bar{y}-y_i)^2$ to be the error of algorithm in section j and the whole cost of the algorithm to be added up by the error and the storage cost multiplied by a tuning parameter λ, that is $Cost_j(\alpha) = Error_j(\alpha) + \lambda Storage_j(\alpha)$.

We first analyze the error of the proposed algorithms.

Theorem 3.3: Considering Algorithms 1 and 2, the maximum error is a function of m, α and $\bar{y}$, which is given by

$$Error_j(\alpha) \leq (m-\alpha)\bar{y}^2$$

The theorem can be proved by finding out the extreme situations and calculate the errors of them. Due to limited space, the proof is not listed here.

From Theorem 3.3, it is shown that the maximum error in a specific section j is related to λ, m, α and $\bar{y}$. The only value can be changed is α because m and $\bar{y}$ depend on the values, and λ is the presupposed tuning parameter. If the number of sections is set to its maximum, i.e., let $\alpha = m$, we have $Error_j(\alpha) = 0$, which is in line with our intuition. More generally, $Error_j(\alpha)$ decreases when α increases, which means that a larger α leads to a smaller error of the algorithms. However, a larger α also costs more storage. Denote ϕ as the storage needed for each number. For

Algorithm 1, node i should keep the current state of α subsections in section j, so a space of $\alpha\phi$ is needed. Therefore, we obtain a theorem as follows.

Theorem 3.4: For Algorithm 1, the possible maximum error and storage cost make up the upper bound of total cost of the in a specific section j, which is

$$\begin{aligned} Cost_j(\alpha) &\leq (m-\alpha)\bar{y}^2 + \alpha\lambda\phi \\ &= (\lambda\phi - \bar{y}^2)\alpha + m\bar{y}^2 \end{aligned}$$

For Algorithm 2, in each iteration, node i receives proportions in section j from its neighbors with a size of $\alpha\phi$, leading to storage space of $\alpha d_i\phi$ consumed. Therefore, we propose a theorem as follows.

Theorem 3.5: For Algorithm 2, the possible maximum error and storage cost make up the upper bound of total cost of the in a specific section j, which is

$$\begin{aligned} Cost_j(\alpha) \leq & (m-\alpha)\bar{y}^2 + \lambda\alpha d_i\phi \\ = & (\lambda d_i\phi - \bar{y}^2)\alpha + m\bar{y}^2 \end{aligned}$$

From Theorem 3.4 and 3.5, it is shown that whether to choose a larger or a smaller α depends on the tuning parameter λ, unit storage cost ϕ, degree d_i (for Algorithm 2) and value of PDF $\bar{y}$. When d_i, ϕ are expected to be large and $\bar{y}$ is expected to be small, a smaller α is better. On the contrary, when there are small d_i, ϕ and large $\bar{y}$, a larger α is recommended.

IV. Numerical Simulations

In this section, we conduct some numerical simulations to illustrate the correctness and performance of our algorithms.

We suppose that there is an area of $a \times a$, and nodes are randomly placed in this area. All nodes share the same connectivity radius l, i.e. if and only if the distance between two nodes is not larger than l, then they can successfully receive information from each other. We also make sure that the graph is interconnected.

Consider an undirected graph G with $n = 40$, $a = 100$ and $l = 20$. The data in the network are generated following standard normal distribution. Figure 1 shows the difference between the temporary variance obtained by each node and the actual variance of values. It is shown that the difference gradually decreases and finally reaches zero as the number of iterations increases. From the graph we can see that the algorithm asymptotically converges, and after around 300 iterations there is almost no error.

Figure 2 shows the partial maximum and minimum values in the network when running Algorithm 1. The two values come closer to each other, and after 780 iterations, the temporary maximum reaches the minimum, so all values in the network have been obtained. The result accords with Theorem 3.1 that the algorithm converges in $(n-1)\lfloor\frac{n+1}{2}\rfloor$ iterations.

In Fig. 3, we show the result of generic PDF calculating algorithm. Define the error in each node at iteration k to be $e[i](k) = \|\rho[i](k) - \rho[i](\infty)\|_2$. Figure 3 shows the relationship between $e[i](k)$ and k. From the graph, it is shown that as the number of iterations increases, the error

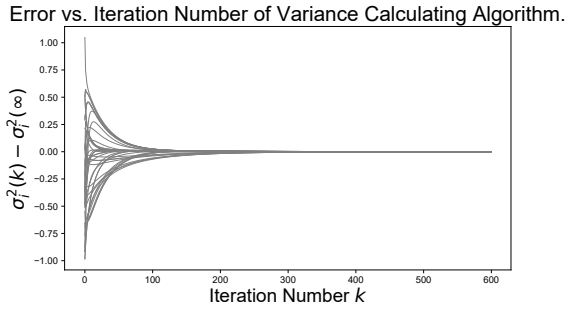

Fig. 1. Error in each node vs. iteration number.

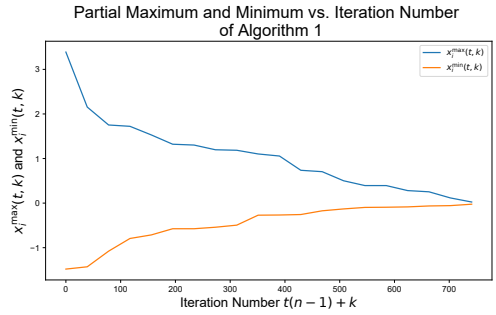

Fig. 2. A typical result of PDF calculating algorithm with IDs.

in each node decreases. The algorithm converges eventually after around 400 iterations with almost zero error.

Figure 4 shows the temporary value of PDF in a specific node when $k = 1, 10$ and 100. It is shown that the distance between the estimate PDF and the actual one gradually decreases with the increase of k.

We also run simulations when keeping n and a unchanged, and choose different l to generate the network topologies, where we make sure that the network is interconnected. Clearly that if l is larger, nodes will be more likely to have more neighbors to exchange information with, i.e. the network has better connectivity. In this simulation, we assume that the diameter d is known to all nodes beforehand. Therefore, algorithm 1 therefore converges in $\lfloor\frac{n+1}{2}\rfloor d$ iterations. We assume that Algorithm 2 has converged in iteration k if and only if all nodes' error $e[i](k)$ is less than 0.025. For each different l, we randomly generate 1000 unique network topologies and record the number of iterations needed for convergence. The result is shown in Fig. 5. It is shown that both algorithms have a larger convergence rate when l increases, i.e. the network has better connectivity. The converges rate of Algorithm 2 is more easily affected by the connectivity than that of Algorithm 1. Therefore, Algorithm 1 is preferred when network connectivity is bad, while Algorithm 2 is preferred when there is good connectivity or IDs are not available in the network.

Figure 6 shows the results when PDF is calculated in 5, 20 or 100 sections. Clearly that the 100 sections one provides the detailed distribution while the 5 sections one shows the roughest distribution. In application, we may choose a value (e.g., 20 in this setting) to balance the accuracy and the

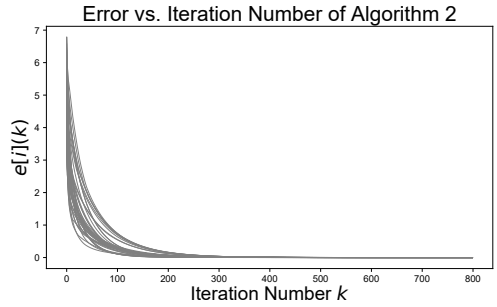

Fig. 3. Error vs. iteration number of Algorithm 2.

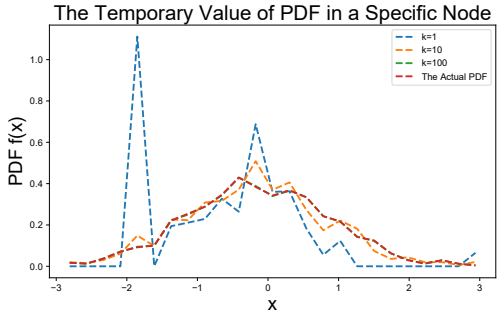

Fig. 4. The temporary PDF in a specific node.

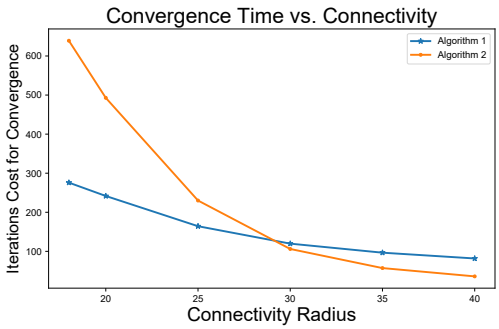

Fig. 5. The relationship between convergence time and connectivity.

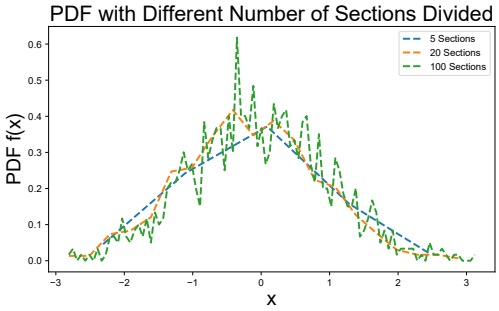

Fig. 6. PDF with different number of sections.

generality.

V. Conclusions

In this paper, we proposed two algorithms to compute the distribution of all values of nodes in a distributed way, considering two network conditions (nodes with or without IDs), respectively. For the nodes with IDs, the PDF of values can be obtained in $O(n^2)$ (or $O(nd)$ when d is known) iterations time with an error of no more than $(m-1)\bar{y}^2$ in each section using the proposed algorithm. For the nodes without IDs, it is proved that the proposed algorithm can asymptotically converge, and has the same convergence rate as average consensus algorithm. Compare with existing algorithms, the proposed algorithms are with low time complexity and can collect a lot more information. We also analyzed the error and storage cost of both algorithms and obtained the upper bounds of them. Simulations were conducted to demonstrate the effectiveness of the proposed algorithms.

References

[1] Chen, M., Mao, S. and Liu, Y., "Big data: A survey". *Mobile Networks and Applications*, 19(2): 171-209, 2014.

[2] Wu, X., Zhu, X., Wu, G. Q., Ding, W. "Data mining with big data". *IEEE Trans. Knowledge and Data Engineering*, 26(1): 97-107, 2014.

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

[4] Reilly, J.P. and Hassett, G.P., Pointcast, Inc. "Information and advertising distribution system and method". U.S. Patent 5,740,549, 1998.

[5] Duan, X., He, J., Cheng, P., Mo, Y., Chen, J. "Privacy preserving maximum consensus". In *Proc. IEEE CDC*, 2015.

[6] J. He, P. Cheng, L. Shi, J. Chen and Y. Sun. "Time synchronization in WSNs: A maximum-value-based consensus approach". *IEEE Trans. on Automatic Control*, 2014, 59(3):660-675.

[7] Olshevsky, A., Tsitsiklis, J. N. "Convergence speed in distributed consensus and averaging". *SIAM Journal on Control and Optimization*, 48(1), 33-55, 2009.

[8] Xiao, L., Boyd, S. "Fast linear iterations for distributed averaging". *Systems and Control Letters*, 53(1), 65-78, 2004.

[9] J. He, P. Cheng, L. Shi, and J. Chen. SATS: Secure average-consensus-based time synchronization in wireless sensor networks. *IEEE Trans. on Signal Processing*, 2013, 61(24): 6387-6400.

[10] Yadav, V., Salapaka, M. V. "Distributed protocol for determining when averaging consensus is reached". *Annual Allerton Conf*, 2007.

[11] J. He, L. Cai, P. Cheng, J. Pan and L. Shi. "Distributed privacy-preserving data aggregation against dishonest nodes in network systems". *IEEE Internet of Things Journal*, DOI:10.1109/JIOT.2018.2834544, 2018

[12] Franceschelli, M., Giua, A. and Pisano, A. "Finite-time consensus on the median value with robustness properties". *IEEE Trans. Automatic Control*, 62(4), pp.1652-1667, 2017.

[13] Liu, H. and Chen, J. "Distributed privacy-aware fast selection algorithm for large-scale data". *IEEE Trans. Parallel and Distributed Systems*, 29(2): 365-376, 2018.

[14] Sun, Y.G., Wang, L. and Xie, G. "Average consensus in networks of dynamic agents with switching topologies and multiple time-varying delays". *Systems and Control Letters*, 57(2), pp.175-183, 2008.

[15] Xiao, L., Boyd, S. and Lall, S. "Distributed average consensus with time-varying metropolis weights". *Automatica*, 2006.

[16] Olfati-Saber, R. and Murray, R.M. "Consensus problems in networks of agents with switching topology and time-delays". *IEEE Trans. Automatic Control*, 49(9): 1520-1533, 2004.

[17] Y. Cao, W. Yu, W. Ren, and G. Chen. "An overview of recent progress in the study of distributed multi-agent coordination". *IEEE Trans. Industrial Informatics*, 9(1): 427-438, 2013.

[18] Sarwate, A.D. and Javidi, T. "Distributed learning of distributions via social sampling." *IEEE Trans. Automatic Control*, 60(1): 34-45, 2015.

[19] Xiao, L., Boyd, S. and Lall, S. April. "A scheme for robust distributed sensor fusion based on average consensus". In *Proc. IEEE IPSN*, 2005.

[20] Iutzeler, Franck et al. "Analysis of max-consensus algorithms in wireless channels." *IEEE Trans. on Signal Processing*, 60 (2012): 6103-6107.