Analysis on relations between graph metrics and average consensus algorithm convergence speed

HUYNH QUANG LONG, University of Twente, The Netherlands

Distributed average consensus is an important problem in the fields of computer science and mathematics, with various applications in AI and IoT. Algorithms exist based on linear algebra, but it is still under investigation how their convergence depends on the graph. Our study analyzed the effects of different graph factors on the convergence time. An experiment with an algorithm on 1000 graphs shows that the diameter, the average distance, the algebraic connectivity, the density, and the max degree are highly correlated to the computation time while the number of vertices is barely related. The average computation time of the algorithm can be approximated as $T = 3.1d^{2.3} = 7.1ad^{2.9} = 168.9ac^{-0.68}$ where d is the diameter, ad is the average distance, and ac is the algebraic connectivity.

Additional Key Words and Phrases: linear system, distributed estimation, networked control, average consensus

1 INTRODUCTION

The distributed consensus problem arises in the context of nodes in a connected network exchanging information with their direct neighbors on agreement or consensus to achieve their common goals. The problem has attracted considerable attention from researchers in recent decades. Among these kinds of problems, average consensus problems have been extensively studied in the computer science literature [1, 9, 10, 14]. In this problem, each node holds a real value, and the nodes' common goal is to find the average of their values. The problem has many applications, including computer networks [8, 11, 12], blockchains [20] and artificial intelligence [13, 17].

An easy way to solve the problem is to let nodes propagate all available information to their neighbors and compute the average value from the known values. The running time is proportional to the longest distance between 2 vertices, making this algorithm's running time optimal. However, the computational cost for exchanges between the nodes is expensive, making it not an ideal choice. Various algorithms with different settings have been introduced. Among them, linear protocols are comprehensively studied due to their low exchange cost and simple implementations [2, 3, 7, 15, 19]. However, analyses of the relations between graph characteristics and the running time of these algorithms are missing or not general. For instance, Gutiérrez-Gutiérrez et al. [7] provide theoretical results for the dependence of the convergence speed of the optimal algorithm on the number of vertices, but only for graphs with very specific structures. This motivates us to research the relation between graph metrics and the convergence speed of average consensus algorithms. In this paper, we use an algorithm proposed by Xiao et al. to [19] discover the following things:

- The factors of graphs that affect the convergence speed of the linear average consensus algorithm.
- Relations between the highly correlated factors and the algorithm's running time.

We chose the algorithm because of its straightforward implementation. The answers will be provided by statistical means. We experimented with the algorithm on 1000 graphs with different metrics to see their relations to the running time. The paper discusses the preliminaries, related work, how we conducted the study, and the experiment results in the following sections. We also reflect on the results and talk about future work in the end.

2 PRELIMINARIES

Consider a connected, undirected graph G with the vertex set V and edge set E. Each vertex in the graph has a value. Define $x \in \mathbb{R}^{|V|}$ as a vector of the vertex values where x_i is the value of vertex i. The vertices want to compute the average of those values by exchanging information with their neighbors. That can be a resource reallocation problem where each agent in the network is assigned a certain amount of work, and they want the work to be evenly distributed. In each iteration, the nodes exchange information with their neighbors and update their values. Denote $x(t)$ as the values of x after t iterations, and $x(0)$ is the initial values of the vertices. During iteration t, the vertices use their information from iteration $t-1$ to calculate and update their value to $x(t)$ according to some predetermined algorithm. The process is converged when $x(t) = \bar{x}$ where $\bar{x}$ is the vector with the average of the values, formally,

$$\bar{x} = \left(\frac{1}{|V|}\sum_{i\in V} x_i\right)\cdot \mathbf{1}$$

where $\mathbf{1} \in \mathbb{R}^{|V|}$ is the vector of ones. We will study an important class of protocols, so-called linear protocols, in which each vertex updates its value based on the weighted sum of its neighbors and itself,

$$x_i(t) = \sum_{j\in V,(i,j)\in E \text{ or } i=j} W_{i,j}x_j(t-1)$$

or,

$$x(t) = Wx(t-1) = W^t x(0)$$

where $W \in \mathbb{R}^{|V|\times|V|}$ is the matrix describing the exchanges between a node and its neighbors. Because vertices only exchange information with their neighbours, we have

$$W_{i,j} = 0 \text{ if } (i,j) \notin E \text{ and } i \neq j.$$

Different W have different convergence speeds on different graphs. However, for a process to converge, it needs that

$$W\bar{x} = \bar{x} \text{ or } W\mathbf{1} = W\mathbf{1}.$$

This condition guarantees that the convergence point is stable. It is also required that

$$\mathbf{1}^T W = W,$$

TScIT 37, July 8, 2022, Enschede, The Netherlands

which means the sum of vertices' values is preserved in each iteration. In this research, we use the optimized W from Xiao et al. [20] that

$$W = \begin{cases} \alpha, & \text{if } i, j \in E \\ 1 - d_i\alpha, & i = j, \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

in which, α is described as

$$\alpha = \frac{2}{\lambda_1(L) + \lambda_{n-1}(L)}$$

where $\lambda_i(L)$ is the i^{th} smallest eigenvalue of the Laplacian matrix [4] of the graph. The algorithm requires graphs to be non-bipartite to guarantee the termination. Additionally, the process may converge asymptotically. Hence, we introduce the termination condition after t iterations as

$$\frac{|x(t) - \bar{x})|}{|x(0)|} < \varepsilon \tag{2}$$

where ε is the error tolerance. The division by the magnitude of the initial vector makes the convergence time independent of the magnitude of the initial values of the vertices, which is defined as

$$T = \min_t \frac{|x(t) - \bar{x})|}{|x(0)|} < \varepsilon \tag{3}$$

In this paper, the terms "convergence time", "number of iterations", and "running time" are used interchangeably.

3 RELATED WORK

In 2004, Xiao et al. proposed optimized W for general graphs as well as an analysis of the problem of finding optimal W [19]. One of the optimized W from the study is used in this research. Another useful study for our research is the recent paper by Gutiérrez-Gutiérrez et al. which provides an optimal W and analysis with formal proofs on the convergence speed for graphs in the form of grid and cycle [7]. On the broader problem of average consensus, Leonidas (2011) proposed a finite average consensus algorithm [5]. The algorithm converges at some point with no error tolerance; however, the convergence rate is slower. A non-linear protocol approach has been seen in the research done by Leonidas and Hasler in 2009, which converges faster in some graphs [6]. In 2019, Xie et al. proposed an algorithm using linear protocols that runs on acyclic graphs with linear complexity [21].

4 METHODOLOGY

The experiment includes 3 stages: creating a graph generator, experimentation, and modeling. The first step of the process is creating a graph generator, which involves choosing the number of vertices and edges, ensuring connectedness and non-bipartiteness. It will then be discussed how we set up the experiment, from choosing possibly related factors to setting the error tolerance ε. Finally, the modeling process is mentioned in the last part of the section, which is about the conditions for a metric to be correlated to the convergence time T and how we express the convergence time T in those factor metrics.

4.1 Graph generator

We need a generator to create graphs and record the characteristics of these graphs. Our graph generator generates graphs with 3 to 200 vertices which are the minimal number of edges to create a connected, non-bipartite graph, and the limit that allows the algorithm to run within a second. To ensure connectedness, we first generate a tree. We create a random Prüfer sequence, from which we build a random tree [16]. The method guarantees that each unique labeled tree with n vertices has an equal probability of being generated. The number of edges m is chosen such that

$$m = n^a$$

where

$$a \sim \mathcal{U}(log_n(n-1), log_n \frac{n*(n-1)}{2}).$$

The selection of m makes factors, such as diameters, large enough and more evenly distributed. We then add an edge to $m - (n-1)$ (the number of edges we need to add to the initial tree) random pairs that are not adjacent in the created tree.

After generating a graph, we need to ensure the non-bipartiteness of the graph. We run a simple depth-first search (DFS). Upon the traversal, if a visited node is not already assigned a group, we assign the node a group that is different from the node we previously visited (the first visited vertex is assigned 0). If the current value of the node is different from the one we previously visited, we go on; otherwise, the graph is not bipartite. If the assignment is possible, the graph is bipartite, and the group of each node is either zero or one. In that case, we select 3 random nodes, 2 of which must have the same group. We then add an edge between them to break the graph bipartiteness.

4.2 Experiment

We first create 1000 graphs using the generator and run the algorithm with each graph. For each run time, we choose the random initial vector x by assigning each vertex in the graph a random value uniformly distributed from 0 to 100. Formally, $x_i(0) \sim \mathcal{U}(0, 100)$. The program runs until it reaches the condition in (2). The ε we chose is $\frac{1}{100}$ which is sufficiently low while keeping the algorithm's running time reasonable. We compute and keep track of the following factors of the graphs:

- Number of vertices.
- Density of the graph, number of edges over the number of vertices
- Algebraic connectivity, the second largest eigenvalue of the Laplacian matrix of the graph.
- Diameter, the length longest shortest path between 2 vertices.
- Average distance, the mean length of the shortest path between 2 vertices.
- Max degree, the maximum size of the incidence set among vertices.

Factors	Correlation
Number of vertices	0.11
Diameter	0.99
Average distance	0.99
Density	-0.84
Algebraic connectivity	-0.98
Max degree	-0.80

Table 1. Correlation values between the metric factors and the number of iterations run in the algorithm.

The density and maximum degree indicate how many channels information can use to travel from one node to another. The convergence rate is anticipated to be slower when these metrics are greater. The greater the number of vertices, the greater the number of vertices that require averaging, which is expected to lengthen the convergence time. The average distance and diameter indicate how long it takes for information to travel between two nodes. With increasing distance, the algorithm's execution time increases. We also examine the graph's connectivity using algebraic connectivity. When the metric is greater, it is anticipated that the running time will be shorter.

4.3 Modeling

In this paper, the correlation between the convergence time T and the metric factors x is referred to as how close we can express T as

$$T \approx cx^a. \tag{4}$$

In other words, we want to see the linear correlation between logarithm values of the factors of a graph and the number of iterations it takes before terminating as

$$\log_2 T \approx a \cdot log_2 x + log_2 c. \tag{5}$$

We then compute the correlation values between the logarithm of the number of iterations and the metrics. If the correlation value p is high ($|p| \geq 0.8$) we plot the regular and log-log graphs for the factor and apply the simple linear regression model to find a and $log_2 c$ in the equation (5) where possible.

5 RESULTS

The generated graphs contain between 3 and 300 vertices, with an average of 152. The number of edges ranges from 3 to 41273, with an average of 2362. The linear correlation between the logarithms of the metrics and the execution time is displayed in the table (1). It can be seen that the diameter, the average distance, the algebraic connectivity, the density, and the max degree are highly related to the number of iterations. Hence, these factors will be looked more into.

5.1 Diameter

Based on the log-log plot in figure (2) and the high correlation value (0.99) in table (1), it appears that logarithms of convergence time and average distance have a strong linear relation. Therefore, we use linear regression to determine a and $log_2 c$ in equation (5), yielding

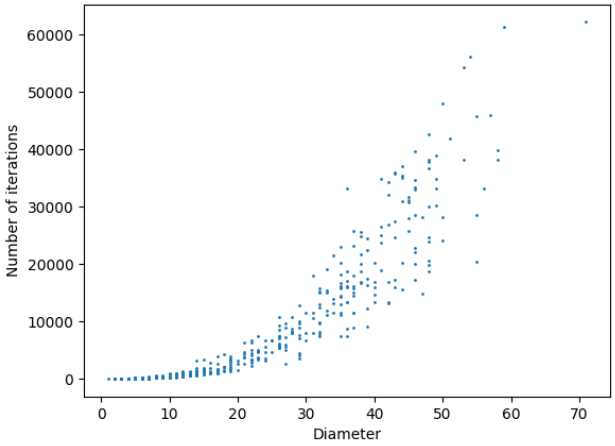

Fig. 1. Relation between the number of iterations and the diameter

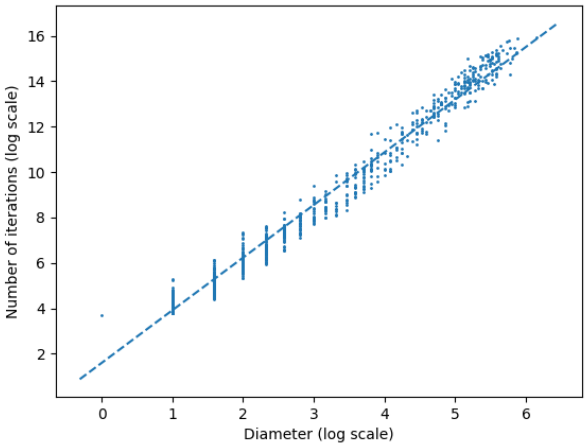

Fig. 2. Relation between the number of iterations and the diameter (logarithms)

$a = 2.3$, $log_2 c = 1.61$, or $c = 3.1$. From that, we get

$$T \approx 3.1 d^{2.3}$$

where d is the diameter of the graph. The computed function corresponds to the shape of points displayed in figure (1). However, the number of iterations demonstrated in the figure varies significantly when the diameter is large.

5.2 Average distance

From log-log plot in the figure (4), and the high correlation value (0.99) in table (1), it suggests a very strong linear correlation between logarithms of the convergence time and the average distance. Hence, we apply the linear regression to find a and $log_2 c$ in equation (5),

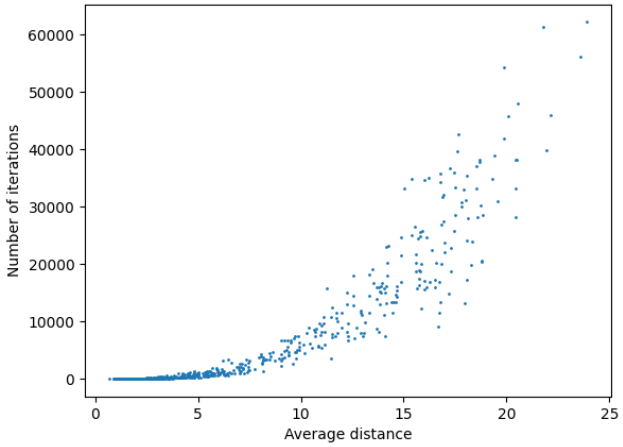

Fig. 3. Relation between the number of iterations and the average distance

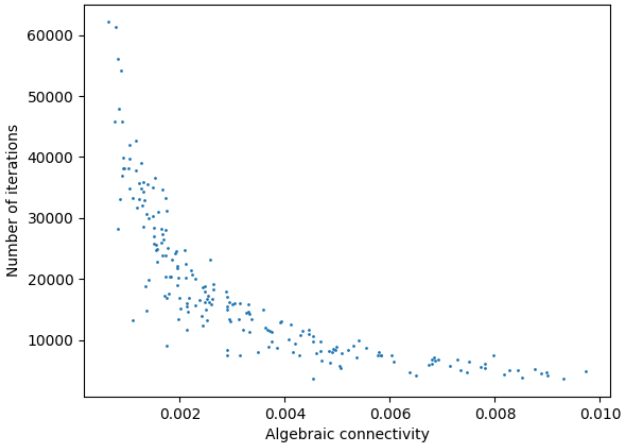

Fig. 5. Relation between the number of iterations and the algebraic connectivity

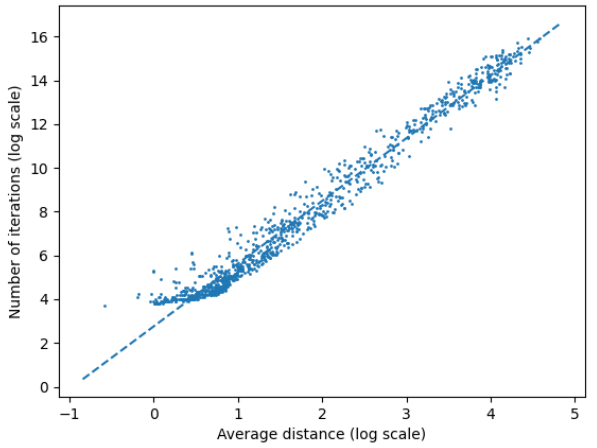

Fig. 4. Relation between the number of iterations and the average distance (logarithms)

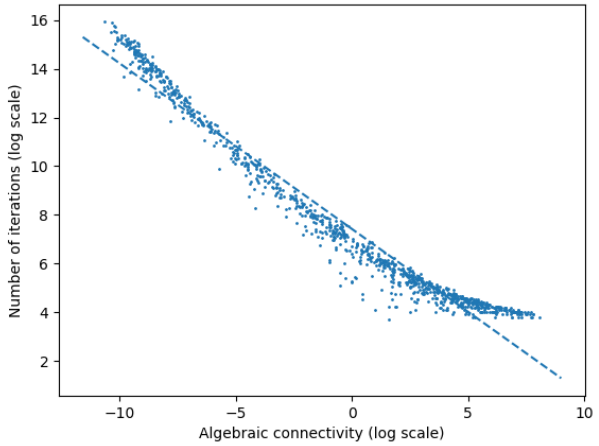

Fig. 6. Relation between the number of iterations and the algebraic connectivity (logarithms)

which yields, $a = 2.9$, $log_2 c = 2.8$, or $c = 7.1$. Thus,

$$T \approx 7.1ad^{2.9}$$

where ad is the average distance of the graph. However, the figure (3) shows that the variance of the number of iterations when the diameter is large is significant.

5.3 Algebraic connectivity

The log-log plot in figure (6) shows a line-like shape. Together with the high correlation in table (1), it suggests that the equation (5) is applicable. Thus, we use the linear regression to find a and $log_2 c$ in the equation, in which we get $a = -0.68$, $log_2 c = 7.1$ implies $c = 168.9$. Hence, we have,

$$T \approx 168.9ac^{-0.68}$$

where ac is the algebraic connectivity of the graph. The regular plot (5) also shows the decaying function similar to the computed one. Due to the high decaying rate, only 500 graphs with the lowest algebraic connectivity are considered in the regular graph (5) to visualize better.

5.4 Density

Although table (1) shows the high correlation value, the plot of logarithm values in figure (8) does not look like a line but more like a decreasing curve. Figure (7) also displays the shape of exponential

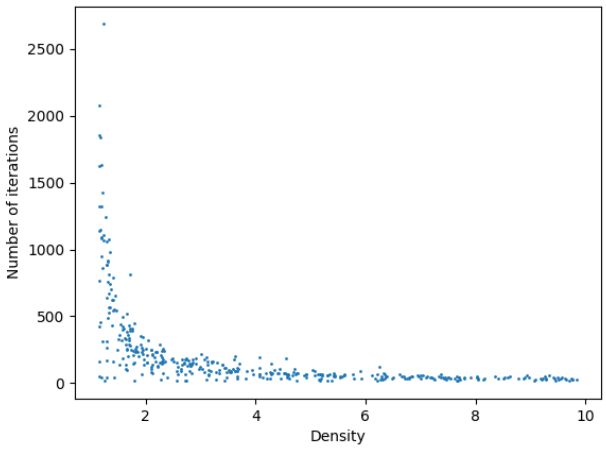

Fig. 7. Relation between the number of iterations and the density

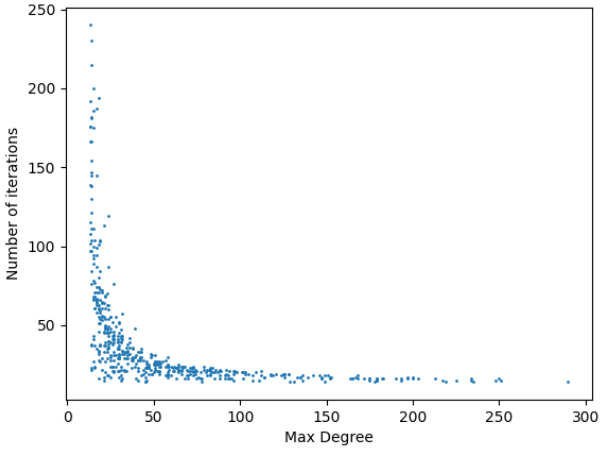

Fig. 9. Relation between the number of iterations and the max degree

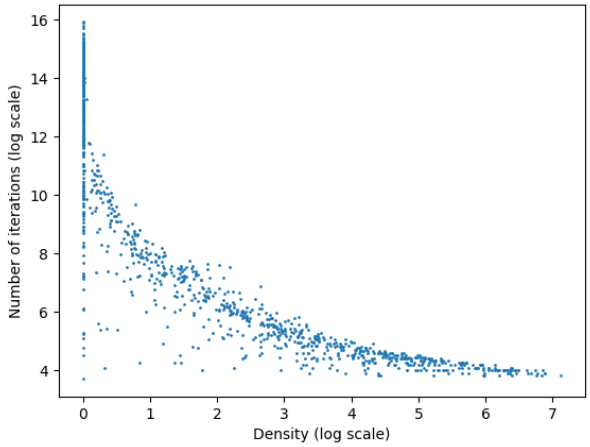

Fig. 8. Relation between the number of iterations and the density (logarithms)

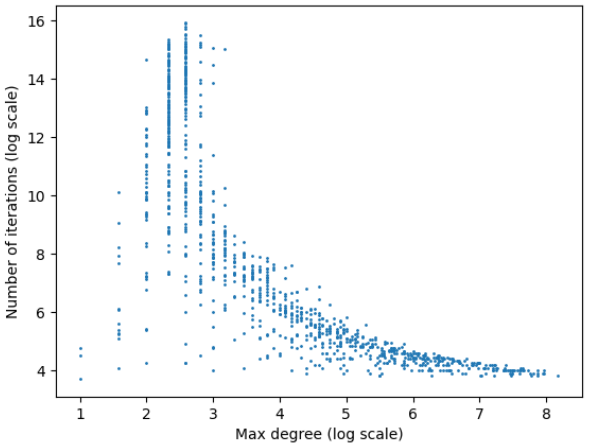

Fig. 10. Relation between the number of iterations and the max degree (logarithms)

decay functions, which does not correspond to equation (4). Hence, linear regression cannot be applied to the factor. Due to the high decaying rate, only 500 graphs with the lowest density are considered in the regular graph (7) for visualization purposes.

5.5 Max degree

Similar to the density, the correlation between max degree and the convergence speed is high, as indicated in the table (1). However, the plot of logarithm values in figure (9) does not demonstrate a linear function. Thus, linear regression cannot be applied in this case. However, it is interesting to research more into the cloud of points in the log-log plot (10), which increases before decreasing, and the decaying function in figure (9).

6 CONCLUSION

Our research has analyzed the relations between the number of iterations of the algorithm and the factors of graphs, such as the number of vertices, diameter, average distance, density, algebraic connectivity, and max degree. For sub-research question 1, the related factors are the average distance, the diameter, the algebraic connectivity, the density, and the max degree while the number of vertices is barely related, as described in table (1). For sub-research question 2, we provide approximating functions $T \approx 3.1d^{2.3}$, $T \approx 7.1ad^{2.9}$, and $T \approx 168.9ac^{-0.68}$ based on diameter d, average distance ad, and algebraic connectivity ac. Although the algorithm's running time is proportional to the average distance cube, the average distance,

in general, is not large with respect to the size of the graph, particularly in the small-world networks, where the average distance is proportional to $ln(n)$ is the number of vertices [18], making the algorithm running time relatively fast.

Our research results indicate that the algorithm's expected running time is $T \approx 3.03d^{2.3}$ where d is the diameter. Compared with the algorithm that runs optimally on grids, lines, and circles, which has the computational complexity $O(d^2)$, there is a small difference between the optimal and the algorithm used in the research. Note that the time complexity from Gutiérrez-Gutiérrez [7] is used for the worst-case running time while ours is the average.

We can see that the effects of diameter and the average distance are similar. Moreover, the correlation value between them is also very high (0.99). That can be either that they are correlated in general or that there are some control factors that create the correlation. In the latter case, our research study has biases.

7 FUTURE WORK

In the data generation part, we did not manage to create a uniformly distributed data set for each factor. That is due to the difficulty of creating unbiased graphs with predefined metrics. Hence, in future research, the distribution of metrics can be made more even.

The study only analyzes the effect of every single factor on the convergence time of the algorithm. However, it can be done with the combinations of 2 or more factors, especially ones with low correlation values with respect to the number of iterations, such as the density and the number of vertices. In addition, future research can work on the non-linear correlation between logarithms of factors such as density and max degree and average consensus algorithms' running times.

Our research did not compute some factors that are possibly related to the algorithm's running time due to difficulty in implementation (e.g., tree-width) or limited computational resources (e.g., NP-hard factors such as the length of the longest cycle or the length of the longest path). That can be an interest for future research.

REFERENCES

[1] Konstantin Avrachenkov, Mahmoud El Chamie, and Giovanni Neglia. 2011. A local average consensus algorithm for wireless sensor networks. In *2011 international conference on Distributed computing in sensor systems and workshops (DCOSS)*. IEEE, 1–6.

[2] Kai Cai and Hideaki Ishii. 2012. Average consensus on general strongly connected digraphs. *Automatica* 48, 11 (2012), 2750–2761.

[3] Ruggero Carli, Fabio Fagnani, Paolo Frasca, and Sandro Zampieri. 2008. A probabilistic analysis of the average consensus algorithm with quantized communication. *IFAC Proceedings Volumes* 41, 2 (2008), 8062–8067.

[4] Fan RK Chung and Fan Chung Graham. 1997. *Spectral graph theory*. Vol. 92. American Mathematical Soc.

[5] Leonidas Georgopoulos. 2011. *Definitive consensus for distributed data inference*. Technical Report. EPFL.

[6] Leonidas Georgopoulos and Martin Hasler. 2009. Nonlinear average consensus. In *Proceedings of the 2009 International Symposium on Nonlinear Theory and its Applications*. IEICE, 10–13.

[7] Jesús Gutiérrez-Gutiérrez, Marta Zárraga-Rodríguez, and Xabier Insausti. 2018. Analysis of known linear distributed average consensus algorithms on cycles and paths. *Sensors* 18, 4 (2018), 968.

[8] Christophe Guyeux, Mohammed Haddad, Mourad Hakem, and Matthieu Lagacherie. 2020. Efficient distributed average consensus in wireless sensor networks. *Computer Communications* 150 (2020), 115–121.

[9] Yue Hao, Yi Li, Xinghua Dong, Li Fang, and Ping Chen. 2018. Performance analysis of consensus algorithm in private blockchain. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 280–285.

[10] Xing He, Junzhi Yu, Tingwen Huang, Chuandong Li, and Chaojie Li. 2018. Average quasi-consensus algorithm for distributed constrained optimization: Impulsive communication framework. *IEEE transactions on cybernetics* 50, 1 (2018), 351–360.

[11] Fangcui Jiang and Long Wang. 2011. Finite-time weighted average consensus with respect to a monotonic function and its application. *Systems & Control Letters* 60, 9 (2011), 718–725.

[12] Shancang Li, George Oikonomou, Theo Tryfonas, Thomas M Chen, and Li Da Xu. 2014. A distributed consensus algorithm for decision making in service-oriented internet of things. *IEEE Transactions on Industrial Informatics* 10, 2 (2014), 1461–1468.

[13] Zhulin Liu and CL Philip Chen. 2017. Broad learning system: Structural extensions on single-layer and multi-layer neural networks. In *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*. IEEE, 136–141.

[14] Hossein Moradian and Solmaz S Kia. 2018. On robustness analysis of a dynamic average consensus algorithm to communication delay. *IEEE Transactions on Control of Network Systems* 6, 2 (2018), 633–641.

[15] Stacy Patterson, Bassam Bamieh, and Amr El Abbadi. 2010. Convergence rates of distributed average consensus with stochastic link failures. *IEEE Trans. Automat. Control* 55, 4 (2010), 880–892.

[16] H Prufer. 1918. Neuer bewis eines satzes uber permutationnen. *Arch. Math. Phys.* 27 (1918), 742–744.

[17] Konstantinos I Tsianos, Sean Lawlor, and Michael G Rabbat. 2012. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *2012 50th annual allerton conference on communication, control, and computing (allerton)*. IEEE, 1543–1550.

[18] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440–442.

[19] Lin Xiao and Stephen Boyd. 2004. Fast linear iterations for distributed averaging. *Systems & Control Letters* 53, 1 (2004), 65–78.

[20] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. 2020. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials* 22, 2 (2020), 1432–1465.

[21] Kan Xie, Qianqian Cai, Zhaorong Zhang, and Minyue Fu. 2019. Distributed algorithms for average consensus of input data with fast convergence. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, 5 (2019), 2653–2664.