23rd International Symposium on Mathematical Theory of Networks and Systems
Hong Kong University of Science and Technology, Hong Kong, July 16-20, 2018

Asynchronous Distributed Parametric Learning with Streaming Data

Ji Liu, Yang Liu, and Wei Chen

***Abstract*— This extended abstract presents an asynchronous algorithm to solve a class of distributed parametric learning problems in a multi-agent network. Each agent acquires its private streaming data to establish a local learning model at times determined by its own clock. It is not assumed that the agents' clocks are synchronized or that the "event times" at which any one agent updates its variables are evenly spaced. The goal is for each agent to converge to a common global learning model, defined as the average of all local ones, by communicating only with its neighbors. It is shown that the algorithm solves the asynchronous distributed parametric learning problems almost surely, or at least with high probability, for any repeatedly jointly strongly connected sequence of neighbor graphs defined on the merged sequence of all agents' event times.**

I. Introduction

Learning, more specifically machine learning, tackles the question of finding a mapping $f : \mathbb{R}^m \to \mathbb{R}$ that takes a feature vector $x \in \mathbb{R}^m$ to a response y, where f often takes either a functional form (e.g., decision tree for classification [1]) or a parametric form (e.g., linear regression [2]). The core idea of learning is to first collect a set of training data in the form of pairs $(x(k), y(k))$, $k \in \{1, 2, \ldots, T\}$, where T is the training size or the number of training samples, and then find an optimal mapping f^* from the set of candidate functions $\mathcal{F}$ that can best describe the data with respect to a certain loss function $l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ (e.g., squared loss $l(z, y) = (z - y)^2$) in the sense that $f^* = \operatorname{argmin}_{f \in \mathcal{F}} \sum_{k=1}^{T} l(f(x(k)), y(k))$. There exist a large body of literature in learning theory and various types of learning; see [3] for details. In this extended abstract, we focus on parametric learning in which f is of a parametric form. Parametric learning techniques have been deployed in many applications ranging from classical artificial intelligence (e.g., image recognition [4] and robotics [5]) to biomedical studies [6] and societal issues [7].

Traditional machine learning relies on a single learning entity (say an agent) to handle all training data, and we thus call it centralized learning, which requires a heavy computation load on the learning agent and may not be applicable in the case when total training data are of huge amount. Due to this scalability issue, decentralized machine learning has been extensively studied in the past few years, in which different approaches have been proposed for distributing training data, and thus computational load, among a group of learning agents (or computational resources). Notable successful solutions include the celebrated ADMM framework [8], the TensorFlow framework [9], the parameter machine based approach [10], and many others [11]–[15]. We call these approaches *decentralized* machine learning, instead of distributed machine learning (as termed in some literature), in order to distinguish the term *distributed* (parametric) learning used in this extended abstract. A key difference between the two learning settings is that in decentralized machine learning, there exists a central agent which collects all the data and distributes them to other agents to leverage parallel computation power, whereas in our setting, there is no such central agent and data are naturally distributed a priori. Such a distributed data collection scheme arises in social network platforms and smart grid power systems in which centralized data collection is not appropriate as data contain private information. Because of this difference, in decentralized machine learning, all agents communicate with the central agent, yet in our setting, each agent only communicates with its (time-varying) neighbors, which will be specified. It has been shown both in theory and practice that learning in a decentralized manner, by distributing data to local computing sources (i.e., agents) and averaging the local learned models from them, achieves comparable results with the corresponding centralized learning [16]–[18]. Notwithstanding this, it is well known that how to reduce communication load between computing sources is a challenging issue in decentralized learning [19].

We consider a more practical scenario that training data arrives sequentially at each agent, which is the same setting as in *online learning*, another recent thread of research in dealing with large scale learning. The streaming property of data motivates the need for learning in an online fashion, as well as the need for learning a model more efficiently when new training samples arrive. These needs stimulate the development of online learning techniques [20]–[24].

In the control community, over the past few decades, there has been significant development in designing algorithms for information distribution and computation among members of a group of agents via peer-to-peer interactions [25]–[27]. Lately, various distributed computation and decision making problems have arisen naturally, such as consensus problems [28], multi-agent coverage problems [29], rendezvous problems [30], and multi-sensor localization [31], because processors onboard sensors or robots are physically separated. Distributed computation and control provide promising settings for large-scale networks because of their desired properties including fault tolerance, cost saving features, and capability to accommodate different physical constraints such as limitations on sensing, computation, and communication.

J. Liu is with the Department of Electrical and Computer Engineering at Stony Brook University (ji.liu@stonybrook.edu). Y. Liu is with the School of Engineering and Applied Sciences at Harvard University (yangl@seas.harvard.edu). W. Chen is with the Department of Electronic & Computer Engineering at The Hong Kong University of Science and Technology (wchenust@gmail.com).

With the preceding discussion in mind, we are interested in distributed parametric learning with streaming data.

In a recent paper [32], a distributed parametric learning problem in a multi-agent network was introduced and studied, in which each agent acquires its private streaming data to establish a local learning model, and the goal is for all agents to learn a common global learning model, defined as the average of all local ones, by communicating only with their neighboring agents. The paper proposes a synchronous algorithm for solving the problem over time-varying undirected graphs, in which the agents only transmit their estimates of the parameter of the global model, but not their private local training data.

There are two limitations of the algorithm in [32]. First, in a realistic sensor or wireless network, it is not always possible to establish *bidirectional* communication between agents. For instance, sensors of different agents may have different sensing radii. Thus, it is more practical to use a directed graph to specify *unidirectional* communications among the agents. Second, the algorithm requires that the updating by all agents are synchronized. For a large wireless network, it is difficult and sometimes impossible to synchronize all agents' clocks [33]. Thus, there is ample motivation to consider an asynchronous version of the problem in which each agent independently updates its variables at times determined by its own clock. What makes the problem asynchronous is that it is *not* assumed that the agents' clocks are synchronized or that the "event times" at which any one agent updates its variables are evenly spaced.

In this extended abstract, we propose an asynchronous algorithm for the distributed parametric learning problem over time-varying unidirectional communications among the agents, which solves the problem almost surely, or at least with high probability, for any repeatedly jointly strongly connected sequence of neighbor graphs defined on the merged sequence of all agents' event times.

II. Problem Formulation

Consider a network consisting of $n > 1$ agents. For the ease of presentation, we label the agents 1 through n. The agents are not aware of such a global labeling, but each agent is able to identify its "neighbors" (which will be defined later). We associate with each agent i a strictly increasing, infinite sequence of *event times* $t_{i0}, t_{i1}, t_{i2}, \dots$ with the understanding that t_{i0} is the time agent i initializes its variables and the remaining t_{ik}, $k > 0$, are the times at which agent i updates its variables. For simplicity and without loss of generality, we assume that $t_{i0} = 0$ for all $i \in \{1, 2, \dots, n\}$. Successive event times of an agent need not be evenly spaced; nor do the event times of one agent have to all be different than the event times of another. We assume that for each $i \in \{1, 2, \dots, n\}$, agent i's event times $t_{i0}, t_{i1}, t_{i2}, \dots, t_{ik}, \dots$ satisfy the constraint

$$\bar{T}_i \geq t_{i(k+1)} - t_{ik} \geq T_i, \qquad k \geq 0, \tag{1}$$

where T_i and $\bar{T}_i$ are positive numbers. An agent i's event times could be any pre-specified sequence of times satisfying the above constraint. The assumption does not preclude arbitrary closeness of event times from different agents' event time sequences. In fact, two agents can have an identical event time.

Each agent observes a sequence of training data $\{x_i(t_{ik}), y_i(t_{ik})\}_k$. Each pair of the observations comes from a local learning model:

$$y_i(t_{ik}) = f_i\left(x_i(t_{ik}), \theta_i\right) + v_i(t_{ik}),$$

where $v_i(t_{ik})$ is some noise, and θ_i is the local parameter to be learnt. We assume that all θ_i's have the same dimension. The goal of the distributed learning problem here is for each agent to learn a global learning parameter θ^*, defined as

$$\theta^* = \frac{1}{n}\sum_{i=1}^{n} \theta_i,$$

in a distributed manner.

For the problem to be solvable, it is clearly necessary that each agent i be able to locally learn θ_i via its private training data $\{x_i(t_{ik}), y_i(t_{ik})\}_k$. We thus impose the following assumption.

Assumption 1: For each agent i, there exists a local learning algorithm

$$\tilde{\theta}_i(t_{ik}) = g_i\left(\{x_i(t_{ip}), y_i(t_{ip})\}_{p=1}^{k}\right)$$

for which $\tilde{\theta}_i(t_{ik})$ converges to θ_i almost surely, or at least with high probability, where by *with high probability* we mean that the probability converges to 1 as $t \to \infty$.

For a single source regression problem, consistency results can often be established for estimating parametric models with i.i.d. training samples; examples include non-regularized and regularized ridge regression [34], non-linear regression models [35], [36], and more general statistical learning [3].

It is worth emphasizing that agents are not allowed to exchange their sampling data because the data may contain private information.

III. The Algorithm

This section presents an asynchronous algorithm which makes use of the idea of double linear iterations [37] (which is also known as push-sum [38], weighted gossip [39], and ratio consensus [40]), and was proposed for solving the distributed averaging problem over directed graphs.

Let $\mathcal{T}_i$ denote the set of all agent i's event times, and $\mathcal{T}$ denote the set of all distinct event times of all n agents. Relabel the elements in $\mathcal{T}$ as $t_1, t_2, \dots, t_l, \dots$ in such a way so that $t_l < t_{l+1}$, $l \geq 1$. It is easy to see that $\mathcal{T}_i$ is an ordered subset of $\mathcal{T}$, and thus there is a positive number which uniformly bounds above the time between any two successive event times in $\mathcal{T}$.

Each agent i has control over three variables, $\tilde{\theta}_i(t)$, $z_i(t)$, and $s_i(t)$, where the updating of $\tilde{\theta}_i(t)$ was introduced in Assumption 1. The variables $\tilde{\theta}_i(t)$ and $z_i(t)$ are agent i's estimates of θ_i and θ^*, respectively. We call $z_i(t)$ and $s_i(t)$ the agreement and scaling variables of agent i, respectively.

In the following algorithm, each agent will send information to its current “out-neighbors” at each of its event times. Consequently, each agent will receive information from its current “in-neighbors” at each of their event times. At each time t, if agent i can send information to agent j, we call agent j an *out-neighbor* of agent i, and agent i an *in-neighbor* of agent j. For each agent i, we use $\mathcal{N}_i^{\rm in}(t)$ and $\mathcal{N}_i^{\rm out}(t)$ to denote the sets of in- and out-neighbors of agent i at time t, respectively, and $n_i^{\rm in}(t)$ and $n_i^{\rm out}(t)$ to denote the number of in- and out-neighbors of agent i at time t, respectively.

In this extended abstract, we assume that there is no communication delay or failure in the network. The algorithm for the asynchronous distributed learning problem is outlined as follows.

Algorithm I: At initial time $t=0$, each agent i initializes its agreement and scaling variables as $z_i(0)=\tilde{\theta}_i(t_{i0})=\tilde{\theta}_i(0)$ and $s_i(0)=1$. At each event time $t\in\mathcal{T}$, the following updating rules apply for each $i\in\{1,2,\ldots,n\}$:

1) If $t\in\mathcal{T}_i$, agent i sends the weighted current values $\frac{z_i(t)}{1+n_i^{\rm out}(t)}$ and $\frac{s_i(t)}{1+n_i^{\rm out}(t)}$ of its agreement and scaling variables to each of its current out-neighbors. Agent i then immediately updates its agreement and scaling variables by setting

$$\begin{aligned} z_i^{\rm new} &= \frac{z_i(t)}{1+n_i^{\rm out}(t)} + \sum_{j\in\mathcal{N}_i^{\rm in}(t)} \frac{z_j(t)}{1+n_j^{\rm out}(t)} \\ &\quad + \tilde{\theta}_i(t+1)-\tilde{\theta}_i(t), \\ s_i^{\rm new} &= \frac{s_i(t)}{1+n_i^{\rm out}(t)} + \sum_{j\in\mathcal{N}_i^{\rm in}(t)} \frac{s_j(t)}{1+n_j^{\rm out}(t)}. \end{aligned}$$

2) If $t\notin\mathcal{T}_i$, agent i updates its agreement and scaling variables by setting

$$\begin{aligned} z_i^{\rm new} &= z_i(t) + \sum_{j\in\mathcal{N}_i^{\rm in}(t)} \frac{z_j(t)}{1+n_j^{\rm out}(t)} \\ &\quad + \tilde{\theta}_i(t+1)-\tilde{\theta}_i(t), \\ s_i^{\rm new} &= s_i(t) + \sum_{j\in\mathcal{N}_i^{\rm in}(t)} \frac{s_j(t)}{1+n_j^{\rm out}(t)}. \end{aligned}$$

It is worth noting that $\mathcal{N}_i^{\rm in}(t)$ may be an empty set. While we do not require updating to happen instantaneously, we do assume that it will be completed before the next event time in $\mathcal{T}$. Agent i then holds these values constant until either it again receives transmission from at least one in-neighbor, or it transmits to its out-neighbors, or both.

To state the main result, we need the following concepts. The communication relationships among the n agents which exist at time $t_l\in\mathcal{T}$ can be described by a directed graph $\mathbb{G}(t_l)$ with vertex set $\mathcal{V}=\{1,2,\ldots,n\}$ and arc set $\mathcal{A}(t_l)\subset\mathcal{V}\times\mathcal{V}$ which is defined in such a way so that for any $i,j\in\{1,2,\ldots,n\}$ and $i\neq j$, the arc from vertex i to vertex j, denoted by (i,j), is an arc in $\mathbb{G}(t_l)$ if and only if $t_l\in\mathcal{T}_i$ and agent j is an out-neighbor of agent i at time t_l. In other words, (i,j) is an arc in $\mathbb{G}(t_l)$ if agent i sends its current weighted agreement and scaling variables to agent j at time t_l. Since each agent's own agreement and scaling variables are always used in its updating, we assume that each $\mathbb{G}(t_l)$ has self-arcs at all n vertices.

A directed graph is called *strongly connected* if there is a directed path between each ordered pair of distinct vertices in the graph. By the *union* of a finite sequence of directed graphs, $\mathbb{G}_1,\mathbb{G}_2,\ldots,\mathbb{G}_p$, each with the same vertex set $\mathcal{V}$, we mean the directed graph with vertex set $\mathcal{V}$ and arc set equaling the union of the arc sets of all of the graphs in the sequence. We say that such a finite sequence is *jointly strongly connected* if the union of its members is a strongly connected graph. We say that an infinite sequence of directed graphs $\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_3,\ldots$, each with the same vertex set, is *repeatedly jointly strongly connected* if there is a positive integer r such that for each integer $k\geq 0$, the finite sequence $\mathbb{G}_{rk+1},\mathbb{G}_{rk+2},\ldots,\mathbb{G}_{r(k+1)}$ is jointly strongly connected.

Theorem 1: Suppose that all n agents adhere to Algorithm I and that the infinite sequence of directed graphs $\mathbb{G}(t_1),\mathbb{G}(t_2),\mathbb{G}(t_3),\ldots$ is repeatedly jointly strongly connected. If each $\tilde{\theta}_i(t)$ converges to θ_i almost surely (or with high probability), then

$$\lim_{t\to\infty} z_i(t)=\theta^*, \qquad i\in\{1,2,\ldots,n\},$$

almost surely (or with high probability).

The proof of this theorem is omitted due to space limitations, and will be given in an expanded version of this extended abstract.

IV. Conclusion

In this extended abstract, we have introduced an asynchronous algorithm to solve a class of distributed parametric learning problems in a multi-agent network, in which each agent acquires its private streaming data to establish a local learning model and aims to learn a common global learning model, defined as the average of all local ones. Without assuming that the agents' clocks are synchronized or that the event times at which any one agent updates its variables are evenly spaced, it has been shown that for any sequence of repeatedly jointly connected graphs describing neighbor relationships defined on the merged sequence of all agents' event times, the algorithm leads all agents to asymptotically converge to the common global learning model, at least with high probability. For future work, it is of interest to study the effects of communication and computation delays, as was done in [41]. Another direction is to further reduce communication load among the agents by utilizing the round robin scheme [37].

References

[1] J. R. Quinlan, “Induction of Decision Trees,” *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[2] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*. John Wiley & Sons, 2015.

[3] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Science & Business Media, 2013.

[4] I. Naseem, R. Togneri, and M. Bennamoun, "Linear Regression for Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 11, pp. 2106–2112, 2010.

[5] M. Dorigo and U. Schnepf, "Genetics-based Machine Learning and Behavior-based Robotics: A New Synthesis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 1, pp. 141–154, 1993.

[6] S. García, A. Fernández, J. Luengo, and F. Herrera, "A Study of Statistical Techniques and Performance Measures for Genetics-based Machine Learning: Accuracy and Interpretability," *Soft Computing*, vol. 13, no. 10, p. 959, 2009.

[7] J. Cohen, P. Cohen, S. West, and L. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Routledge, 2013.

[8] 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, 2011.

[9] M. Abadi *et al.*, "TensorFlow: A System for Large-scale Machine Learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, 2016.

[10] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems," *arXiv preprint arXiv:1512.01274*, 2015.

[11] F. J. Provost and D. N. Hennessy, "Scaling Up: Distributed Machine Learning with Cooperation," in *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 74–79, 1996.

[12] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, "Map-Reduce for Machine Learning on Multicore," in *Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS)*, pp. 281–288, 2006.

[13] M. Lauer and M. Riedmiller, "An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems," in *Proceedings of the 17th International Conference on Machine Learning*, pp. 535–542, 2000.

[14] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling Up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011.

[15] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.

[16] Y. Zhang, J. C. Duchi, and M. J. Wainwright, "Communication-Efficient Algorithms for Statistical Optimization," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 3321–3363, 2013.

[17] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip Learning with Linear Models on Fully Distributed Data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 556–571, 2013.

[18] C. Huang and X. Huo, "A Distributed One-Step Estimator," *arXiv preprint arXiv:1511.01443*, 2015.

[19] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication Efficient Distributed Machine Learning with the Parameter Server," in *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*, pp. 19–27, 2014.

[20] M. Zinkevich, *Online Convex Programming and Generalized Infinitesimal Gradient Ascent*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2003.

[21] Y. Wang and S. Boyd, "Fast Model Predictive Control Using Online Optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[22] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, 2010.

[23] J. Gittins, K. Glazebrook, and R. Weber, *Multi-armed Bandit Allocation Indices*. John Wiley & Sons, 2011.

[24] S. Shalev-Shwartz, "Online Learning and Online Convex Optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[25] J. N. Tsitsiklis, *Problems in Decentralized Decision Making and Computation*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 1984.

[26] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[27] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip Algorithms for Distributed Signal Processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[28] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[29] J. Cortés, S. Martínez, T. Karataş, and F. Bullo, "Coverage Control for Mobile Sensing Networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[30] J. Lin, A. S. Morse, and B. D. O. Anderson, "The Multi-agent Rendezvous Problem. Part 1: the Synchronous Case," *SIAM Journal on Control and Optimization*, vol. 46, no. 6, pp. 2096–2119, 2007.

[31] L. Hu and D. Evans, "Localization for Mobile Sensor Networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, pp. 45–57, 2004.

[32] J. Liu, Y. Liu, A. Nedić, and T. Başar, "An approach to distributed parametric learning with streaming data," in *Proceedings of the 56th IEEE Conference on Decision and Control*, pp. 3206–3211, 2017.

[33] N. M. Freris, S. R. Graham, and P. R. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1352–1364, 2011.

[34] R. Cummings, S. Ioannidis, and K. Ligett, "Truthful Linear Regression," in *Proceedings of the 28th Conference on Learning Theory (COLT)*, pp. 448–483, 2015.

[35] B. P. Rao, "The Rate of Convergence of the Least Squares Estimator in a Non-Linear Regression Model with Dependent Errors," *Journal of Multivariate Analysis*, vol. 14, no. 3, pp. 315–322, 1984.

[36] G. Lebanon, *m-Estimators and z-Estimators*. Citeseer.

[37] J. Liu and A. S. Morse, "Asynchronous distributed averaging using double linear iterations," in *Proceedings of the 2012 American Control Conference*, pp. 6620–6625, 2012.

[38] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pp. 482–491, 2003.

[39] F. Bénézit, V. Blondel, P. Thiran, J. N. Tsitsiklis, and M. Vetterli, "Weighted gossip: distributed averaging using non-doubly stochastic matrices," in *Proceedings of the 2010 IEEE International Symposium on Information Theory*, pp. 1753–1757, 2010.

[40] A. D. Domínguez-García, S. T. Cady, and C. N. Hadjicostis, "Decentralized optimal dispatch of distributed energy resources," in *Proceedings of the 51st IEEE Conference on Decision and Control*, pp. 3688–3693, 2012.

[41] J. Liu, S. Mou, and A. S. Morse, "Asynchronous Distributed Algorithms for Solving Linear Algebraic Equations," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 372–385, 2018.