DEEP LEARNING-BASED AVERAGE CONSENSUS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we study the problem of accelerating the linear average consensus algorithm over complex networks. We specifically present a data-driven methodology for tuning the weights of temporal (i.e., time-varying) networks by using deep learning techniques. We first unfold the linear average consensus protocol to obtain a feedforward signal flow graph, which we regard as a neural network. We then train the neural network by using standard deep learning technique to minimize the consensus error over a given finite time-horizon. As a result of the training, we obtain a set of optimized time-varying weights for faster consensus in the complex network. Numerical simulations are presented to show that our methodology can achieve a significantly smaller consensus error than the static optimal strategy.

1 INTRODUCTION

The distributed agreement problem on networks, often referred to as a consensus problem (Olfati-Saber et al., 2007), is an important problem in the network science and engineering, with applications in load balancing (Cybenko, 1989), data fusion (Xiao et al., 2005), multi-agent coordination (Ren & Beard, 2005), distributed computing (Xiao & Boyd, 2004), distributed sensor networks (Cortés & Bullo, 2005), wireless communication systems (Senel & Akar, 2017), and power systems (Dörfler & Bullo, 2012). Recently, it also appears in online machine learning procedures to process big data (Chen & Sayed, 2012; Tsianos et al., 2012, and references therein).

In the *average* consensus problem, nodes in the network seek to converge their state variables to the average of their initial states in a distributed manner. The standard solution to the average consensus problem is to use the linear average consensus algorithm (Olfati-Saber & Murray, 2004), in which each node updates its state by taking a weighted linear average of its own state and the state of its neighbors. This algorithm results in a linear dynamical system whose state transition matrix involves the Laplacian matrix of the underlying communication network.

Designing consensus algorithms with fast convergence speed is of significant practical interest because such algorithms allow the multi-agent systems to reach an agreement with fewer iterations and, therefore, by consuming less communication resource. In the context of the linear average consensus algorithm, the problem of finding the optimal weights of edges for maximizing the asymptotic consensus speed can be reduced to a convex optimization problem (Xiao & Boyd, 2004), under the assumption that the communication network is static and undirected. It was recently shown by Kempton et al. (2018) that the optimal weights can be computed in a distributed manner by an iterative computation. Zelazo et al. (2013) clarified the role of cycles in the linear average consensus algorithm and presented a methodology for accelerating the consensus by adding edges to a network. On the other hand, for the case of directed networks, Hao & Barooah (2012) presented a method to accelerate the convergence rate of a linear (but not necessarily an average) consensus algorithm by tuning the weights of edges in the network.

A natural consequence of seeking for further acceleration of consensus algorithms is the emergence of finite-time consensus algorithms (Sundaram & Hadjicostis, 2007), in which edge-weights are typically assumed to be time-varying and the designer exploits the additional flexibility to realize consensus in a finite-time. The finite-time consensus algorithm proposed by Hendrickx et al. (2015) achieves consensus by stochastic (but possibly asymmetric) matrices in $N(N-1)/2$ iterations, where N denotes the number of nodes in the network. Safavi & Khan (2015); Shang (2016) used tools from graph signal processing (see, e.g., Shuman et al. (2013)) to show that, by allowing non-stochasticity

for the state-update matrices, one can realize a finite-time consensus in at most N steps. The theoretical aspects of these works have been further investigated by Apers & Sarlette (2017). Recently, Falsone et al. (2018) showed that the number of steps required for consensus can be further improved to $N/2$ in the specific case of ring networks having an even number of nodes.

Despite the aforementioned advances for consensus acceleration, there is still a lack of an effective methodology for answering the following basic question: Given a finite time-window as well as an underlying network structure, how should we dynamically tune the edge weights in the network for achieving as accurate consensus as possible at the end of the time-window? If the length of the time-window is not long enough to run the aforementioned finite-time consensus algorithms, currently available options are effectively limited to using the static optimal strategies (e.g., Xiao & Boyd (2004)), which does not allow us to dynamically tune the weights of the network. To fill in this gap, in this paper we present a data-driven approach for tuning the weights of an undirected temporal (i.e., time-varying) networks by using deep learning techniques. We first unfold the consensus algorithm and obtain a feedforward signal flow graph (Ito et al., 2019), which we regard as a neural network. We then use the standard stochastic gradient descent algorithm to train the parameters in each layer of the neural network (i.e., the weights of each snapshot of the temporal network) to minimize the consensus error over a finite time-horizon, which results in an optimized temporal network for faster consensus. We numerically confirm that our approach can drastically accelerate the convergence speed in the linear average consensus algorithm.

This paper is organized as follows. In Section 2, we state the problem of dynamically tuning the edge weights to accelerate the linear average consensus algorithm, and then we propose our methodology for solving the problem using standard techniques in the field of deep learning. In Section 3, we evaluate the performance of the proposed method with various numerical simulations. We finally conclude the paper in Section 4.

2 Weight Optimization by Deep Learning Techniques

In this section, we describe our methodology for tuning the edge weights of the networks for accelerating the linear average consensus algorithm within a given finite-time horizon. We first give a brief review of the linear average consensus algorithm and state its basic properties. We then describe our data-driven methodology for tuning the weights of the network, in which we apply the techniques in the deep learning to the signal flow graph obtained by unfolding the consensus algorithm.

2.1 Linear average consensus algorithm

Let G be an undirected and unweighted network having the node set $V = \{1,\dots,N\}$ and the edge set E consisting of unordered pairs of nodes in V. Each node in G represents an agent, which is supposed to communicate with its neighbors at each time. In this paper, we focus on the discrete-time case. Let $x_i(k) \in \mathbb{R}$ denote the state of the ith node at time $k \geq 0$, and $\mathcal{N}_i$ denote the set of neighbors of node i. In the standard linear average consensus protocol (Olfati-Saber et al., 2007), each node i updates its own state according to the following difference equation:

$$x_i(k+1) = x_i(k) + \sum_{j\in\mathcal{N}_i} w_{ij}(k)(x_j(k) - x_i(k)), \;\; x_i(0) = x_{0,i}, \tag{1}$$

where $w_{ij}(k) = w_{ji}(k) \geq 0$ represents the weight of the (undirected) edge $\{i,j\}$ at time k and $x_{0,i}$ is the initial state of node i. For each time $k \geq 0$, we define the (i,j) element of the adjacency matrix of the network $W(k) \in \mathbb{R}^{N\times N}$ by

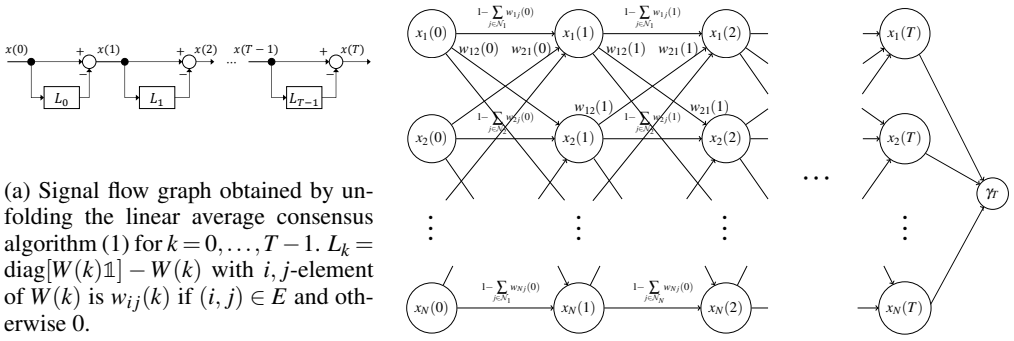
$$W_{ij}(k) = \begin{cases} w_{ij}(k), & \text{if } j \in \mathcal{N}_i, \\ 0, & \text{otherwise,} \end{cases}$$

and the degree matrix of the network at time k by

$$D(k) = \mathrm{diag}(d_1(k),\dots,d_N(k)), \;\; d_i(k) = \sum_{j\in\mathcal{N}_i} w_{ij}(k).$$

Then, using the Laplacian matrix of the network

$$L(k) = D(k) - W(k),$$



(a) Signal flow graph obtained by unfolding the linear average consensus algorithm (1) for $k = 0, \ldots, T-1$. $L_k = \mathrm{diag}[W(k)\mathbb{1}] - W(k)$ with i,j-element of $W(k)$ is $w_{ij}(k)$ if $(i,j) \in E$ and otherwise 0.

(b) Corresponding neural network

Figure 1: Unfolded signal flow graph and corresponding neural network

the evolution of the state vector

$$x(k) = [x_1(k) \quad \cdots \quad x_N(k)]^\top$$

in the linear average consensus protocol (1) is written as

$$x(k+1) = (I - L(k))x(k), \;\; x(0) = x_0,$$

where

$$x_0 = [x_{1,0} \quad \cdots \quad x_{N,0}]^\top$$

denotes the initial state vector.

The objective of this paper is to present a framework for tuning the weights $\{w_{ij}(k)\}_{k\geq 0,\{i,j\}\in E}$ for the faster average consensus in a given finite time window. Let us denote the average of the initial states of the nodes by

$$c = \frac{1}{N}\sum_{i=1}^{N} x_{i,0}.$$

Define the consensus error vector

$$e(k) = x(k) - c\mathbb{1},$$

where $\mathbb{1}$ denotes the all-one N-dimensional column vector. We are now ready to state the problem studied in this paper.

Problem 2.1 (Consensus acceleration problem)**.** *Let G be an undirected and unweighted network having N nodes. Let T be a positive integer. Assume that the set of initial states follow a probability distribution $\mathcal{X}_0$, i.e.,*

$$\{x_{0,1}, \ldots, x_{0,N}\} \sim \mathcal{X}_0.$$

Find the set of nonnegative weights

$$\{w_{ij}(k)\}_{k\in\{0,1,\cdots,T-1\},\{i,j\}\in E}$$

that minimizes the average consensus error defined by

$$\gamma_T = E[\|e(T)\|],$$

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^N$, and $E[\cdot]$ denotes the expected value.

Because Problem 2.1 is a non-convex problem, it is difficult to compute a set of $\{w_{ij}(k)\}_{k\geq 0,\{i,j\}\in E}$ that minimizes γ_T. This difficulty motivates us to tackle this problem using a data-driven approach to find a suboptimal solution. In the next subsection, we describe our data-driven approach for tuning the edge weights by using deep leaning techniques. We remark that, although we assume our knowledge of the initial probability distribution $\mathcal{X}_0$ in the process of optimization, the optimized edge-weights can drastically accelerate the consensus protocol even if the initial states do not follow the given distribution. We numerically illustrate this universality property of our approach in Subsection 3.4.

2.2 DATA-DRIVEN WEIGHT OPTIMIZATION

To adjust the weights by using deep learning techniques, we first unfold the recursive state-update formula (1) and obtain a signal-flow graph shown in Fig. 1a. Unlike a standard deep neural network, the resulting neural network has a structure and contains no activation function. The structure of the neural network corresponds to the structure of the graph G, and the same between all layers. The neurons of kth layer corresponds to the nodes at time k as shown in Fig. 1b.

We then apply a standard technique in the field of deep learning to adjust the weights. We use the mean squared error, γ_T^2, as the loss function, which is then regularized by appending a regularization term of the Frobenius norm. As in Ito et al. (2019), we use the technique of the incremental training for adjusting the weights during the training process. In the incremental training, we first consider only the first layer (i.e., we set $k = 1$ in Fig. 1a) and attempt to minimize the regularized loss function of the average consensus error $\gamma_1^2 + \lambda \|W(0)\|_F$ using a number of randomly generated initial state x_0 as the training data, which we call the 1st generation. After training the first set of weights $w_{ij}(0)$, we proceed to training the first two sets of edge weights by appending the second layer to the neural network and replacing the loss function by $\gamma_2^2 + \lambda \|W(1)\|_F$. In this training, we use the result from the 1st generation as the initial value of the first layer and train the entire neural network. We repeat this process to finally optimize the weights $w_{ij}(T-1)$ between $T-1$st and Tth layers by minimizing $\gamma_T^2 + \lambda \|W(T-1)\|_F$. We train the network with a stochastic gradient descent algorithm.

3 PERFORMANCE EVALUATION

In this section, we illustrate the effectiveness of the proposed method by various numerical simulations. The weighting factor for the regularization is set to $\lambda = 1$, the number of data-set per learning is set to 10000, and the size of minibatch is one. For evaluations, 100 samples are used. With the above setup, the simulations were performed in PyTorch (Paszke et al., 2017) using Adam with learning rate 0.01 for training.

3.1 BASELINE STRATEGY

Throughout this section, we compare the performance of the proposed method with that of the static optimal strategy presented in Xiao & Boyd (2004). Assume that the initial state x_0 is a deterministic vector. Let us further assume that the edge weights $w_{ij}(k)$ do not depend on time k. Under these assumptions, Xiao & Boyd (2004) have shown that the problem of finding the static edge weights minimizing the (worst-case) asymptotic convergence factor

$$r_{\mathrm{asym}} = \sup_{x_0 \neq c\mathbb{1}} \limsup_{k\to\infty} \left(\frac{\|e(k)\|}{\|e(0)\|} \right)^{1/k} \tag{2}$$

reduces to solving a linear matrix inequality, which can be globally and efficiently solved (Boyd et al., 1994). Then, as the baseline strategy, we use the following time-invariant consensus protocol

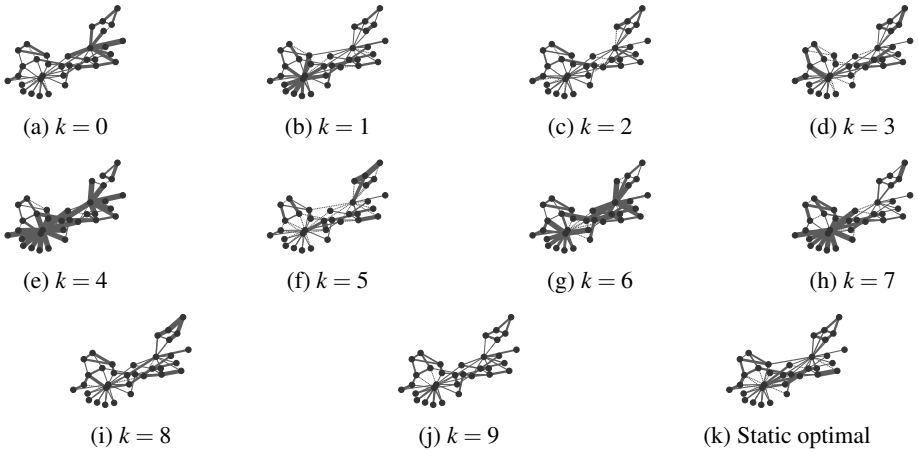
$$x_i(k+1) = x_i(k) + \sum_{j\in\mathcal{N}_i} w_{ij}^{\mathrm{stat}}(x_j(k) - x_i(k)), \;\; 0 \leq k \leq T-1, \tag{3}$$

where w_{ij}^{stat} are the static optimal weights obtained by solving the linear matrix inequality.

3.2 DETERMINISTIC NETWORKS

In this subsection, we use the following two empirical and synthetic deterministic networks; Karate network (Zachary, 1977) ($N = 34$ nodes) and the square lattice network ($N = 6^2 = 36$ nodes). We assume that the initial state of each node independently follows a uniform distribution on the interval $[-1, 1]$.

For Karate network, we set $T = 10$ and numerically optimized the edge weights of the network at the times $k = 0, \ldots, 9$. In Fig. 2, we present the optimized weights of edges in the network. We then empirically evaluated the average consensus error $E[\|e(k)\|]$ for $k = 0, \ldots, 9$. The results are shown in Fig. 3. The accuracy of the consensus achieved by the proposed method is about 50 times better than the static optimal policy. We observe that the optimized weights of the network are



(a) $k = 0$ (b) $k = 1$ (c) $k = 2$ (d) $k = 3$

(e) $k = 4$ (f) $k = 5$ (g) $k = 6$ (h) $k = 7$

(i) $k = 8$ (j) $k = 9$ (k) Static optimal

Figure 2: Optimized weight of edges in the Karate network. (a)–(j): Proposed method. (k): Static optimal strategy. The width of the lines indicate the values of the weights (the thicker a line is, the larger its weight is). The edges having weight less than 10^{-2} are indicated by dashed lines.

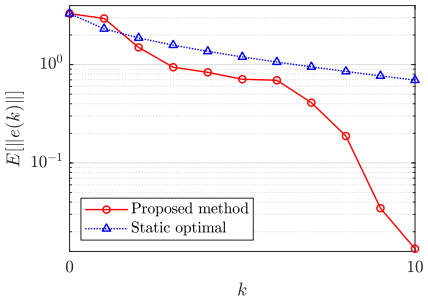
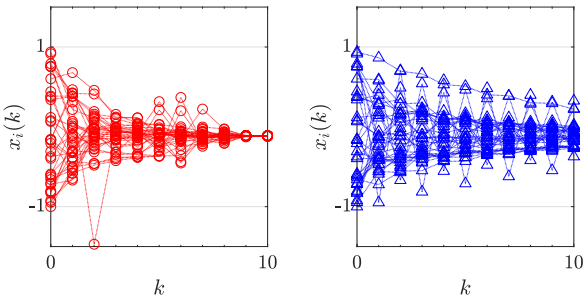


Figure 3: Empirical mean of the consensus errors (Karate network)



(a) Proposed method (b) Static optimal

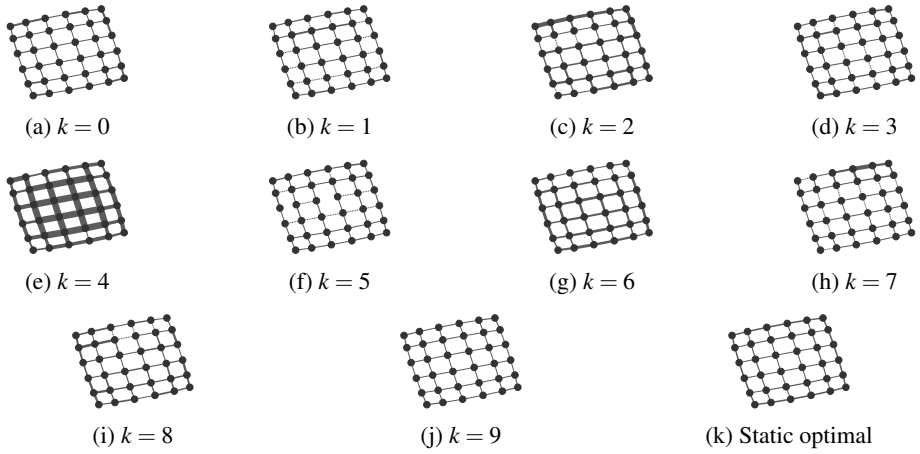
Figure 4: State trajectories (Karate network)

dynamically changing in a non-trivial manner. In Fig. 4, we show the sample trajectories of the average consensus protocol with the static optimal and proposed edge weights. We confirm that the proposed edge weights achieves a more precise average consensus at the final time $k = 10$ compared with the static optimal strategy.

We then consider the average consensus on the square lattice network. We set $T = 10$ and numerically optimized the edge weights of the network. The results are shown in Figs. 5–7. The accuracy of the consensus by the proposed method at the final time $k = 10$ is about 14 times better than the static optimal methodology. The optimized weights show a trend similar to the one for the Karate network. However, it is worth noting that the weights at time $k = 4$ are relatively large at various edges in the network. This sudden increase in edge weights in fact drives the nodes away from the consensus state but only temporarily. After all, despite this phenomena, the proposed approach allows the nodes to achieve a better average consensus at the final time.

3.3 Random synthetic networks

We consider the following three random and synthetic network models: the Erdős-Rényi (ER) network ($N = 100$ nodes and $M = 252$ edges, where the probability for edge creation is 0.05) the Barabási-Albert (BA) model (Barabási & Albert, 1999) ($N = 100$ and $M = 291$, where the number of edges to attach from a new node to existing nodes is 3), and the Watts-Strogatz (WS) model (Watts & Strogatz, 1998) ($N = 100$ and $M = 200$, where each node is joined with its 4 nearest neighbors in a ring topology, and the probability of rewiring each edge is 0.15). We set $T = 10$. As in the case of the deterministic networks, we assume that the initial states of the nodes independently follow a uniform distribution on the interval $[-1, 1]$. For each of the networks, we used the deep learning technique to find the weights of the edges at times $k = 0, \ldots, 9$. We then evaluated the empirical



(a) $k = 0$ (b) $k = 1$ (c) $k = 2$ (d) $k = 3$

(e) $k = 4$ (f) $k = 5$ (g) $k = 6$ (h) $k = 7$

(i) $k = 8$ (j) $k = 9$ (k) Static optimal

Figure 5: Optimized weight of edges in the lattice network. (a)–(j): Proposed method. (k): Static optimal strategy. The width of the lines indicate the values of the weights (the thicker a line is, the larger its weight is). The edges having weight less than 10^{-2} are indicated by dashed lines.

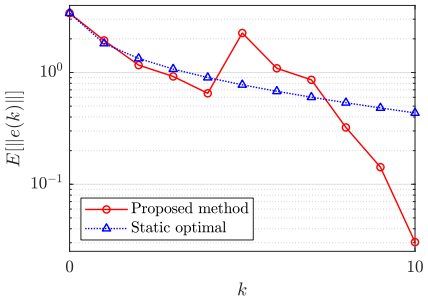
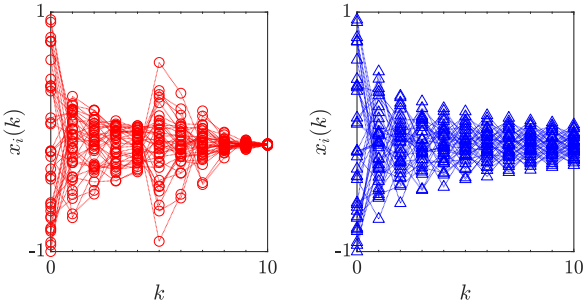
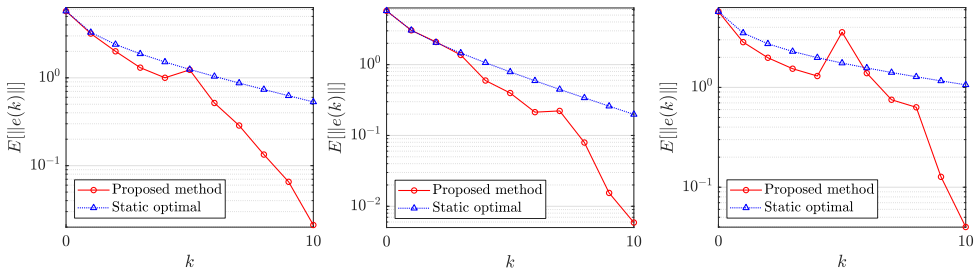


Figure 6: Empirical mean of the consensus errors (lattice network)



(a) Proposed method (b) Static optimal

Figure 7: State trajectories (lattice network)



(a) Erdős-Rényi network (b) Barabási-Albert model (c) Watts-Strogatz model

Figure 8: Mean consensus errors for random network models.

average of the consensus error $E[\|e(k)\|]$ for $k = 0, \ldots, 10$. We show the results in Fig. 8. As in the case of the deterministic networks in Subsection 3.2, the proposed method achieves significantly less consensus errors at the final time.

We notice that, only in the case of the WS network, the consensus error temporarily and significantly increases at time $k = 5$. In order to examine if this phenomena is specific to the WS model, the following experiment was performed: For each of the three random graph models, we created 10 realizations of networks, for which we ran the proposed algorithm to obtain the optimized edge weights. We then empirically computed the mean consensus errors $E[\|e(k)\|]$ for $k = 0, \ldots, 10$ for each of the 3×10 cases. We show the results in Fig. 9. From the figure, we confirm that only the case of WS network model presents a temporal increase in the mean consensus errors, while the errors from the other two cases decrease almost monotonically.

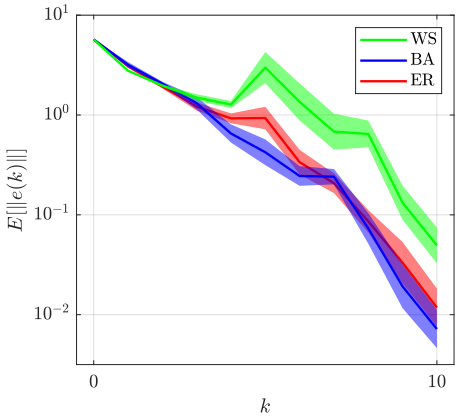


Figure 9: Distribution of the mean consensus errors for the ER, BA, and WS network models. Solid lines and shaded areas represent the averages and the standard deviations, respectively.

Table 1: Asymptotic convergence factors

	Proposed	Baseline
Karate	6.69×10^{-1}	9.25×10^{-1}
Lattice	7.23×10^{-1}	9.21×10^{-1}
ER	6.53×10^{-1}	8.70×10^{-1}
BA	5.85×10^{-1}	7.79×10^{-1}
WS	6.99×10^{-1}	9.35×10^{-1}

3.4 Periodic continuation

In the previous subsections, we have confirmed that the proposed method can drastically accelerate the average consensus algorithm under the assumption that we are given a prespecified finite time-window and that we know the distribution of the initial states. In this subsection, we further show that a periodic continuation of our algorithm discussed in Subsection 2.2 yields an consensus algorithm that effectively accelerates the consensus for *any* initial state vector and over an *infinite* time-window.

For given G and T, let $L^\star(0), \ldots, L^\star(T-1)$ denote the optimized weighted Laplacian matrices of the network. Then, the consensus algorithm proposed in Subsection 2.2 is written as

$$x(k+1) = (I - L^\star(k))x(k), \quad 0 \le k \le T-1.$$

By periodically extending the state transition matrices $I - L^\star(0), \ldots, I - L^\star(T-1)$, we obtain the following average consensus protocol over an infinite time horizon:

$$x(sT+\tau+1) = \left(\prod_{t=0}^{\tau}(I - L^\star(\tau - t))\right)x(sT), \;\; 0 \le \tau \le T-1, \; s \ge 0. \tag{4}$$

The next lemma gives an explicit representation of the asymptotic convergence factor (2) of the consensus algorithm (4).

Lemma 3.1. *Let $L^\star(0), \ldots, L^\star(T-1)$ denote the optimized weighted Laplacian matrices of the networks by our deep learning algorithm. The asymptotic convergence factor of the consensus algorithm (4) equals*

$$r_{\mathrm{asym}}^T = \sup_{x_0 \neq c\mathbb{1}} \limsup_{k\to\infty} \left(\frac{\|e(k)\|}{\|e(0)\|}\right)^{1/k} = \left\|\prod_{t=0}^{T-1}(I - L^\star(T-1-t))\right\|^{1/T}, \tag{5}$$

where $\|\cdot\|$ is the spectral norm.

Proof. First note that

$$e(k+1) = \prod_{t=0}^{k}(I - L^\star(k-t))e(0).$$

By expressing k using s and τ as

$$k = sT + \tau, \;\; \tau \in \{0, \cdots, T-1\},$$

we have

$$\limsup_{k\to\infty} = \limsup_{s\to\infty} \max_{\tau\in\{0,\cdots,T-1\}}$$

because $\mathcal{T} := \{0,\cdots,T-1\}$ is a finite countable set. Next, notice that

$$\begin{aligned}
r_{\text{asym}}^T &= \sup_{x_0\neq c\mathbb{1}} \limsup_{s\to\infty} \max_{\tau\in\mathcal{T}} \left(\frac{\|x(sT+\tau+1)-c\mathbb{1}\|}{\|e(0)\|}\right)^{\frac{1}{sT+\tau+1}} \\
&= \sup_{x_0\neq c\mathbb{1}} \limsup_{s\to\infty} \max_{\tau\in\mathcal{T}} \left(\frac{\|\prod_{t=0}^{sT+\tau}(I-L^\star(sT+\tau-t))e(0)\|}{\|e(0)\|}\right)^{\frac{1}{sT+\tau+1}} \\
&= \limsup_{s\to\infty} \max_{\tau\in\mathcal{T}} \left\|\prod_{t=0}^{sT+\tau}(I-L^\star(sT+\tau-t))\right\|^{\frac{1}{(sT+\tau+1)}} \\
&\leq \limsup_{s\to\infty} \max_{\tau\in\mathcal{T}} \left(\left\|\prod_{t=0}^{\tau-1}(I-L^\star(\tau-1-t))\right\| \left\|\left(\prod_{t=0}^{T-1}(I-L^\star(T-1-t))\right)^s\right\|\right)^{\frac{1}{sT+\tau+1}} \\
&\leq \limsup_{s\to\infty} \max_{\tau\in\mathcal{T}} \left\|\prod_{t=0}^{T-1}(I-L^\star(T-1-t))\right\|^{\frac{s}{sT+\tau+1}} = \left\|\prod_{t=0}^{T-1}(I-L^\star(T-1-t))\right\|^{\frac{1}{T}}.
\end{aligned}$$

Here, we used

$$\limsup_{s\to\infty} \gamma^{1/s} = 1$$

for a constant γ. On the other hand, we have

$$\begin{aligned}
r_{\text{asym}}^T &\geq \sup_{x_0\neq c\mathbb{1}} \limsup_{s\to\infty} \left(\frac{\|x(sT)-c\mathbb{1}\|}{\|e(0)\|}\right)^{\frac{1}{sT}} \\
&= \limsup_{s\to\infty} \left\|\left(\prod_{t=0}^{T-1}(I-L^\star(T-1-t))\right)^s\right\|^{\frac{1}{sT}} = \left\|\prod_{t=0}^{T-1}(I-L^\star(T-1-t))\right\|^{\frac{1}{T}}.
\end{aligned}$$

This completes the proof. □

Lemma 3.1 states that r_{asym}^T is the geometric mean of the spectral norm of the product of T-step transition matrices of $I-L^\star(T-1-t)$, $t=0,\cdots,T-1$. Note that r_{asym}^T with $T=1$ does not necessarily correspond to the static-optimal in (2).

Using Lemma 3.1, we computed the asymptotic convergence factor of the consensus algorithm (4) with $T=10$ for each of the five networks (i.e., Karate, lattice, ER, BA, and WS networks). We also computed the asymptotic convergence factor of the baseline strategy (3) for each of the five networks. The obtained asymptotic convergence factors are given in Table 1. We observe that the proposed method achieved less convergence factors, which shows the effectiveness of the proposed approach even in the case of infinite time-horizon problems.

4 Conclusion

In this paper, we have presented a data-driven approach for accelerating the linear average consensus algorithm over undirected temporal networks. We have first unfolded the consensus algorithm to obtain an equivalent feedforward signal flow graph, which we have regarded as a neural network. We have then showed that we can apply standard deep learning techniques to train the obtained neural network and obtain a temporal network having optimized edge-weights. We have numerically confirmed that our methodology can outperform the average consensus algorithm with the static optimal edge-weights.

REFERENCES

Simon Apers and Alain Sarlette. Accelerating consensus by spectral clustering and polynomial filters. *IEEE Transactions on Control of Network Systems*, 4(3):544–554, 2017.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial Mathematics, 1994.

Jianshu Chen and Ali H. Sayed. Diffusion adaptation strategies for distributed optimization and learning over networks. *IEEE Transactions on Signal Processing*, 60(8):4289–4305, 2012.

Jorge Cortés and Francesco Bullo. Coordination and geometric optimization via distributed dynamical systems. *SIAM Journal on Control and Optimization*, 44(5):1543–1574, 2005.

George Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7(2):279 – 301, 1989.

Florian Dörfler and Francesco Bullo. Synchronization and transient stability in power networks and nonuniform Kuramoto oscillators. *SIAM Journal on Control and Optimization*, 50(3):1616–1642, 2012.

Alessandro Falsone, Kostas Margellos, Simone Garatti, and Maria Prandini. Finite-time distributed averaging over gossip-constrained ring networks. *IEEE Transactions on Control of Network Systems*, 5(3):879–887, 2018.

He Hao and Prabir Barooah. Improving convergence rate of distributed consensus through asymmetric weights. In *American Control Conference*, pp. 787–792, 2012.

Julien M. Hendrickx, Guodong Shi, and Karl H. Johansson. Finite-time consensus using stochastic matrices with positive diagonals. *IEEE Transactions on Automatic Control*, 60(4):1070–1073, 2015.

Daisuke Ito, Satoshi Takabe, and Tadashi Wadayama. Trainable ISTA for sparse signal recovery. *IEEE Transactions on Signal Processing*, 67(12):3113–3125, 2019.

Louis Kempton, Guido Herrmann, and Mario Di Bernardo. Self-organization of weighted networks for optimal synchronizability. *IEEE Transactions on Control of Network Systems*, 5(4):1541–1550, 2018.

Reza Olfati-Saber and Richard M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *31st Conference on Neural Information Processing Systems*, 2017.

Wei Ren and Randal W. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, 2005.

Sam Safavi and Usman A. Khan. Revisiting finite-time distributed algorithms via successive nulling of eigenvalues. *IEEE Signal Processing Letters*, 22(1):54–57, 2015.

Kamil Senel and Mehmet Akar. A distributed coverage adjustment algorithm for femtocell networks. *IEEE Transactions on Vehicular Technology*, 66(2):1739–1747, 2017.

Yilun Shang. Finite-time weighted average consensus and generalized consensus over a subset. *IEEE Access*, 4(8):2615–2620, 2016.

David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

Shreyas Sundaram and Christoforos N Hadjicostis. Finite-time distributed consensus in graphs with time-invariant topologies. In *American Control Conference*, pp. 711–716, 2007.

Konstantinos I. Tsianos, Sean Lawlor, and Michael G. Rabbat. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *Annual Allerton Conference on Communication, Control, and Computing*, pp. 1543–1550, 2012.

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

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

Lin Xiao, Stephen Boyd, and Sanjay Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Fourth International Symposium on Information Processing in Sensor Networks*, pp. 63–70, 2005.

Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

Daniel Zelazo, Simone Schuler, and Frank Allgöwer. Performance and design of cycles in consensus networks. *Systems and Control Letters*, 62(1):85–96, 2013.