Distributed computation of fast consensus weights using ADMM

Kiran Rokade and Rachel Kalpana Kalaimani

***Abstract*—We consider the problem of achieving average consensus among multiple agents, where the inter-agent communication network is depicted by a graph. We consider the discrete-time consensus protocol where each agent updates its value as a weighted average of its own value and those of its neighbours. Given a graph, it is known that there exists a set of 'optimal weights' such that the agents reach average consensus asymptotically with an optimal rate of convergence. However, existing methods require the knowledge of the entire graph to compute these optimal weights. We propose a method for each agent to compute its set of optimal weights locally, i.e., each agent only has to know who are its neighbours. The method is derived by solving a matrix norm minimization problem subject to linear constraints in a distributed manner using the Alternating Direction Method of Multipliers (ADMM). We illustrate our results using numerical examples and compare our method with an existing method called the Metropolis weights, which are also computed locally.**

***Index Terms*—Consensus algorithm, distributed optimization, ADMM.**

I. Introduction

In a consensus problem, a group of agents seeks to agree upon a certain quantity of interest. In many applications, this quantity of interest is the average of the initial states of the agents [1], [2]. Consensus problems arise in various settings such as multi-agent formation control [3], estimating a parameter using a network of sensors [1], workload balancing for distributed computation [2], etc. These problems have been widely studied in control theory [4]–[7].

In all consensus protocols, an important parameter is the time required to reach average consensus. One line of literature focuses on algorithms which achieve consensus in finite time [8]–[10]. The other line of literature looks at algorithms which achieve consensus asymptotically, while trying to improve the rate of this asymptotic convergence [6], [11]. For continuous-time consensus protocols, it is known that the rate of convergence depends on the second-smallest eigenvalue of the graph Laplacian [5], while for weighted discrete-time protocols, it depends on the second-largest absolute value of all eigenvalues of the weight matrix [6]. It is then natural to optimize this rate of convergence by appropriately choosing the edge and node weights used by the agents in their update laws for reaching consensus.

This work has been partially supported by DST-INSPIRE Faculty Grant, Department of Science and Technology (DST), Govt. of India (ELE/16-17/333/DSTX/RACH).

Kiran Rokade is with the Department of Electrical and Computer Engineering, Cornell University and Rachel Kalpana Kalaimani is with the Department of Electrical Engineering, Indian Institute of Technology Madras (e-mail: kvr36@cornell.edu, rachel@ee.iitm.ac.in)

The problem of optimizing the rate of convergence of distributed linear discrete-time consensus protocols has been widely studied in literature and is sometimes referred to as the fastest distributed linear averaging (FDLA) problem [6]. In [6], the authors consider the general weighted discrete-time consensus protocol, with possibly asymmetric edge weights. They propose two metrics for quantifying the rate of convergence: the asymptotic convergence factor and the per-step convergence factor. It is shown that the problem of optimizing the per-step convergence factor is a convex optimization problem and hence can be solved efficiently. In [11], the authors consider consensus protocols with symmetric edge weights. Considering the edge and node weights as variables, they try to minimize the distance between the largest and the second-smallest eigenvalue of the weighted graph Laplacian. This in-turn optimizes the rate of convergence of the consensus protocol. In [12], the authors consider the problem of maximizing the second-smallest eigenvalue of the graph Laplacian where the edge weights are functions of the agent values. They propose an iterative method which gives optimal trajectories of agent values by solving an optimization problem at each iteration. Another class of problems which falls under the FDLA framework is that of fastest mixing Markov chains [13]. Here, the equation governing the evolution of the probability distribution is similar to the discrete-time consensus protocol. The objective is to find a transition probability matrix, analogous to the weight matrix in a consensus protocol, such that an initial probability distribution converges to a uniform distribution as fast as possible.

While the above methods give weights that achieve the optimal rate of convergence in a consensus protocol, finding these weights require the knowledge of the entire network topology. In other words, there has to be a central entity which has the knowledge of the entire network, can calculate the optimal weights and then broadcast them to the agents. Firstly, there may not exist such a central entity due to constraints on the communication resources or for security reasons. Secondly, even if it does exist, this process of computing and broadcasting the weights has to be repeated every time the network topology changes due to say node failure, edge failure or if new agents enter the network. This can be very inefficient if the network is large. In the spirit of distributed computation, we would want the agents to locally compute these weights which they use in their update laws. An example of locally computed weights is the Metropolis weights [1]. To compute them, an agent has to know only the degree of itself and its neighbours. While these weights do achieve consensus, the rate of convergence of the consensus protocol with these weights is not optimal. We seek to have the best of both worlds: locally computed weights whose rate of convergence is optimal.

A solution to the problem of locally computing the optimal consensus weights has been attempted in [14]. There, the authors use a distributed gradient-descent approach on the Schatten p-norm of the weight matrix, which is an approximation of the second-largest eigenvalue of the weight matrix. Computing the gradient requires the agents to communicate with its neighbours which are up to $p/2$ hops away. There is a trade-off between locality and optimality: smaller values of p give sub-optimal solutions. We propose an algorithm where the agents need to communicate only with their immediate neighbours to compute the consensus weights. Moreover, the rate of convergence of the consensus protocol using the weights obtained by our algorithm can be arbitrarily close to the optimal value. The algorithm is derived using the Alternating Direction Method of Multipliers (ADMM) [15], [16]. We summarize our contributions below.

A. Contributions

1) We consider agents which communicate with each other over a network depicted by an undirected graph. We propose an iterative algorithm for the agents to locally compute the weights required for the weighted-average discrete-time consensus protocol. We show that the rate of convergence of the consensus protocol using these weights converges to the optimal value. The algorithm is derived by solving a matrix norm minimization problem in a distributed manner using ADMM. The result is given in Section III.
2) For some applications, we propose a variation of our algorithm, where, at every iteration, the agents compute the weights and update their values using these weights. Using a numerical example, we show that this variation of our algorithm performs better in terms of the rate of convergence of the consensus protocol than the locally computed Metropolis weights. This is done in Section IV-B.

The rest of the paper is organized as follows. We next summarize some basic notations which will be used throughout the paper. In Section II, we mathematically formulate the problem of computing the optimal consensus weights. In Section III, we propose our algorithm to locally compute these optimal weights. We also state our main result which proves the convergence of this algorithm. The proof of the result is deferred to the appendix. In Section IV, we present some numerical examples to illustrate the convergence of our algorithm and to compare its performance with the Metropolis weights. Finally, we give some concluding remarks in Section V.

B. Notation

All vectors are of length n and all matrices are of size $n\times n$. $\mathbf{1}$ is the vector $\begin{bmatrix}1 & \dots & 1\end{bmatrix}^T$ of all ones. e_i is the i^{th} standard basis vector with 1 as the i^{th} entry and zeros elsewhere. For a vector x: x_i or $(x)_i$ denotes its i^{th} element, $||x||$ denotes its 2-norm. For a matrix A: $\rho(A)$ denotes its spectral radius or the maximum absolute eigenvalue, $||A||$ denotes its induced 2-norm or the maximum singular value, $||A||_F$ denotes its Frobenius norm, A_{ij} or $(A)_{ij}$ denotes its $(i,j)^{\text{th}}$ element, $\mathrm{Tr}(A)$ denotes its trace, $A>0$ denotes $A_{ij}>0$ for all (i,j). For a set B, $|B|$ denotes its cardinality. For a function $f:\mathbb{R}^n\to\mathbb{R}$, $(\partial/\partial x)f=\begin{bmatrix}(\partial/\partial x_1)f & \dots & (\partial/\partial x_n)f\end{bmatrix}^T$ denotes the gradient of f with respect to the vector $x\in\mathbb{R}^n$. For a function $f:\mathbb{R}^{n\times n}\to\mathbb{R}$,

$$(\partial/\partial X)f=\begin{bmatrix}(\partial/\partial X_{11})f & \dots & (\partial/\partial X_{1n})f\\ \vdots & \ddots & \vdots\\ (\partial/\partial X_{n1})f & \dots & (\partial/\partial X_{nn})f\end{bmatrix}$$

denotes the gradient of f with respect to the matrix $X\in\mathbb{R}^{n\times n}$.

II. Problem Formulation

A. The average consensus problem

Consider a set $V=\{1,\dots,n\}$ of n agents, each having a scalar initial value $x_i(0)\in\mathbb{R}, i\in\{1,\dots,n\}$. Assume that the inter-agent communication is governed by an undirected, connected graph $G=(V,E)$, where E is the set of all undirected edges of G. In other words, agents i and j can exchange values with each other if and only if $(i,j)\in E$. Note that an agent always knows its own value, hence, for the ease of notation, we assume that $(i,i)\in E$ for all $i\in\{1,\dots,n\}$. Let $N_i=\{j\in V:(i,j)\in E\}$ be the set of all neighbours of agent i.

We want the agents to reach average consensus, i.e., each agent must compute the average $x_{\text{avg}}(0)=(1/n)\sum_{i=1}^n x_i(0)$ of the initial values in a distributed manner by communicating only with its neighbours. We consider the distributed linear iterative protocol where each agent i updates its value as

$$x_i(t+1)=W_{ii}x_i(t)+\sum_{j\in N_i}W_{ij}x_j(t),\ i\in\{1,\dots,n\},\quad (1)$$

where $t\in\{0,1,2,\dots\}$ is the time and W_{ij} is the weight assigned by agent i to agent j's value. Due to the communication constraint imposed by the graph, we fix $W_{ij}=0$ if $(i,j)\notin E$.

Remark 1. Note that since the graph G is undirected, $(i,j)\in E$ implies W_{ij} and W_{ji} can both be nonzero. However, they can take different values in general.

Now, the protocol (1) can be written as

$$x(t+1)=Wx(t),\quad (2)$$

where $W\in\mathbb{R}^{n\times n}$ is called the *weight matrix* and $x(t)=\begin{bmatrix}x_1(t) & \dots & x_n(t)\end{bmatrix}^T\in\mathbb{R}^n$ is the vector of agent values at time t. We refer to (2) as the *consensus protocol*. Let $\bar{x}=x_{\text{avg}}(0)\mathbf{1}$ be the vector with $x_{\text{avg}}(0)$ as all its entries. From (2), we have $x(t)=W^tx(0)$. Thus, the agents reach average consensus, i.e., $\lim_{t\to\infty}x(t)=\bar{x}$ if and only if

$$\lim_{t\to\infty}W^t=\mathbf{1}\mathbf{1}^T/n.\quad (3)$$

Following result from [6] gives a necessary and sufficient condition for (3) to be true.

Proposition 1. *[6] The consensus protocol* (2) *reaches average consensus, i.e.,* $\lim_{t\to\infty} x(t) = \bar{x}$ *if and only if*

$$W\mathbf{1} = \mathbf{1},\ W^T\mathbf{1} = \mathbf{1},\ \rho(W - \mathbf{1}\mathbf{1}^T/n) < 1. \tag{4}$$

Note that the weights W_{ij} are allowed to be negative in general. Condition (4) ensures that in the consensus protocol given by (2):

- the average $\mathbf{1}^T x(t)/n$ of the agent values is invariant across time t,
- any vector in the 'agreement space', i.e., in the span of $\mathbf{1}$, is invariant with respect to W,
- all eigenvalues of W other than the eigenvalue 1 are strictly inside the unit circle.

Given that W satisfies (4), Proposition 1 guarantees $\lim_{t\to\infty} x(t) = \bar{x}$. We are interested in maximizing the rate at which this convergence occurs.

B. Optimal weights

Define $e(t) = x(t) - \bar{x}$ as the consensus error. Then, from the consensus protocol (2), we can write the dynamics of the consensus error as

$$e(t+1) = (W - \mathbf{1}\mathbf{1}^T/n)e(t). \tag{5}$$

Now, one way to characterize the rate of convergence of the consensus protocol is by the spectral radius $\rho(W - \mathbf{1}\mathbf{1}^T/n)$. This quantity is the largest absolute eigenvalue of W other than the eigenvalue 1. It is called the asymptotic convergence factor [6]. The weight matrix W which gives the best asymptotic convergence factor can be found by solving the optimization problem

$$\begin{aligned} &\text{minimize} && \rho(W - \mathbf{1}\mathbf{1}^T/n) \\ &\text{subject to} && W\mathbf{1} = \mathbf{1},\ W^T\mathbf{1} = \mathbf{1}, \\ &&& W_{ij} = 0, (i,j) \notin E, \end{aligned} \tag{P1}$$

where the constraint

$$W_{ij} = 0, (i,j) \notin E \tag{6}$$

is the topological constraint imposed by the graph G. It is known that $\rho(W - \mathbf{1}\mathbf{1}^T/n)$ is in general a non-convex function of W and hence (P1) is a hard problem to solve [17].

We replace the objective function $\rho(W - \mathbf{1}\mathbf{1}^T/n)$ in (P1) by the convex function $||W - \mathbf{1}\mathbf{1}^T/n||$. By definition of the induced matrix norm, we have

$$||W - \mathbf{1}\mathbf{1}^T/n|| = \sup_{e(t)\neq 0, t\geq 0} \frac{||e(t+1)||}{||e(t)||}, \tag{7}$$

i.e., $||W - \mathbf{1}\mathbf{1}^T/n||$ captures the worst case one-step increase in the consensus error. Hence, $||W - \mathbf{1}\mathbf{1}^T/n||$ is known as the per-step convergence factor [6]. Now, consider the convex problem

$$\begin{aligned} &\text{minimize} && ||W - \mathbf{1}\mathbf{1}^T/n|| \\ &\text{subject to} && W\mathbf{1} = \mathbf{1},\ W^T\mathbf{1} = \mathbf{1}, \\ &&& W_{ij} = 0, (i,j) \notin E. \end{aligned} \tag{P2}$$

We denote a solution of (P2) by W^*. Note that since $W \neq W^T$ in general (refer Remark 1), we have $\rho(W - \mathbf{1}\mathbf{1}^T/n) \leq ||W - \mathbf{1}\mathbf{1}^T/n||$. We show that if the graph G is connected, then W^* satisfies $\rho(W^* - \mathbf{1}\mathbf{1}^T/n) < 1$.

Lemma 1. *The weight matrix W^* obtained by solving the convex problem* (P2) *satisfies the average consensus condition given in* (4).

The proof of Lemma 1 is given in Appendix C. Henceforth, we refer to $||W - \mathbf{1}\mathbf{1}^T/n||$ as simply the *convergence factor* of W. Thus, W^* is a weight matrix which achieves consensus and gives an optimal convergence factor for the consensus protocol. Due to the topological constraint $W_{ij} = 0, (i,j) \notin E$, solving (P2) requires the knowledge of the entire graph G. Thus, (P2), in its original form, can only be solved in a 'centralized manner'. We later propose to solve the problem in a distributed manner using ADMM.

C. Locally calculated weights

Distributed algorithms require that each agent computes the average $x_{\text{avg}}(0)$ using only local information from its neighbours. In the same spirit, we would want each agent i to calculate its optimal weights $\{{W^*}_{ij}, j = 1,\dots,n\}$ locally, without the knowledge of the entire graph. Before looking at the problem of locally computing the optimal weights, we look at a set of weights, which, although not optimal, can be computed locally.

Consider the set of weights called the local degree weights or the Metropolis weights. We denote the matrix of these weights by W_{M}. For each agent $i \in \{1,\dots,n\}$, let

$$(W_{\text{M}})_{ij} = \begin{cases} 0, & (i,j) \notin E, \\ \min\left\{\frac{1}{1+|N_i|}, \frac{1}{1+|N_j|}\right\}, & (i,j) \in E, i \neq j, \\ 1 - \sum_{j\in N_i}(W_{\text{M}})_{ij}, & i = j. \end{cases} \tag{8}$$

Intuitively, with these weights used in the consensus protocol, the information of an agent carries more weight if it has few neighbours. This should speed up the propagation of the information of the not-so-well-connected agents in the network, which will improve the overall rate of convergence of the entire network. These weights are widely used since they are simple to calculate and can be shown to achieve average consensus [1]. However, as we shall see later through a numerical example, the convergence factor of W_{M} can be significantly poor than W^*. Our aim is to give a method for computing weights locally, whose convergence factor is optimal.

Problem 1. Given an undirected graph $G = (V, E)$, derive a method such that each agent $i \in \{1,\dots,n\}$ can determine its set of weights $\{W_{ij}, j = 1,\dots,n\}$ knowing only its neighbour set N_i such that the weight matrix W

1) satisfies the average consensus condition given in (4) and
2) has an optimal convergence factor, i.e.,

$$||W - \mathbf{1}\mathbf{1}^T/n|| = ||W^* - \mathbf{1}\mathbf{1}^T/n||,$$

where W^* is a solution of (P2).

We propose a solution to Problem 1 by solving (P2) in a distributed manner. We do this using the well-known

Alternating Direction Method of Multipliers (ADMM) [16]. It gives fairly accurate results in relatively fewer number of iterations than other methods [15]. Due to this, it has been widely used in solving practical optimization problems, e.g. [18], [19]. Our result is presented in the next section.

III. Main Result

In this section, we propose our result which shows how (P2) can be solved in a distributed manner over an undirected, connected graph. First, consider the problem

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \frac{||W_i - \mathbf{1}\mathbf{1}^T/n||}{n} \qquad \text{(P3)} \\ \text{subject to} \quad & W_i = W_j, (i,j) \in E, \\ & W_i\mathbf{1} = \mathbf{1},\ W_i^T\mathbf{1} = \mathbf{1},\ i \in \{1,\dots,n\}, \\ & (W_i)_{ij} = 0, (i,j) \notin E, i \in \{1,\dots,n\}. \end{aligned}$$

Here, we have replaced the W in (P2) by n matrices $W_1,\dots,W_n$, one for each agent in the network. The matrix W_i can be seen as agent i's estimate of the centralized optimal solution W^*. Since G is connected, the constraint $W_i = W_j, (i,j) \in E$ implies all W_i's are equal. This implies (P3) is equivalent to (P2). Introducing these new matrices will enable solving (P3) in a distributed manner. A key step towards this is to impose the topological constraint $(W_i)_{ij} = 0, (i,j) \notin E$ only on the i^{th} row of the matrix W_i, for each $i \in \{1,\dots,n\}$. This means each agent can impose this constraint on its estimate W_i knowing only its neighbour set N_i. We show that (P3) can be solved in a distributed manner. The steps are given in Algorithm 1. The result is formally stated below, its proof can be found in Appendix A.

Theorem 1. *Consider the n sequences of matrices $\{W_i(k)\}_{k\geq 0}, i \in \{1,\dots,n\}$, where each sequence is generated by running Algorithm 1 in parallel on each agent $i \in \{1,\dots,n\}$. Then, in the limit as k goes to infinity, all n matrices $W_i(k), i \in \{1,\dots,n\}$*

1) *are equal, i.e.,*
$$\lim_{k\to\infty} ||W_i(k) - W_j(k)|| = 0$$
for all $i,j \in \{1,\dots,n\}$,
2) *satisfy the constraints of problem* (P2) *and*
3) *give an optimal convergence factor for the consensus protocol* (2)*, i.e.,*
$$\lim_{k\to\infty} ||W_i(k) - \mathbf{1}\mathbf{1}^T/n|| = ||W^* - \mathbf{1}\mathbf{1}^T/n||$$
for all $i \in \{1,\dots,n\}$, where W^* is a solution of the centralized problem (P2).

It is known that for a convex optimization problem having only equality constraints, the ADMM iterates converge to the optimal objective value and satisfy the constraints of the optimization problem in the limit as the number of iterations goes to infinity [15, Section 3.2.1]. Hence, to prove Theorem 1, all we need to show is that Algorithm 1 is an ADMM implementation of problem (P3). This is done in Appendix A. We now make some remarks on the algorithm.

Algorithm 1 Calculating locally an estimate of the optimal weight matrix W^* at agent i of the network G

```
Initialize: $\rho > 0, W_i(0) = 0_{n\times n}, a_i(0) = 0_{n\times 1}, b_i(0) = 0_{n\times 1}, M_i(0) = 0_{n\times n}$
Send $W_i(0)$ to all neighbours $j \in N_i$.
Receive $W_j(0)$ from all neighbours $j \in N_i$.
Iterate:
for $k \geq 0$ do
    Primal update:
    Evaluate $W_i(k+1)$ as per (13).
    Exchange values:
    Send $W_i(k+1)$ to all neighbours $j \in N_i$.
    Receive $W_j(k+1)$ from all neighbours $j \in N_i$.
    Dual update:
        $a_i(k+1) = a_i(k) + \rho(W_i(k+1)\mathbf{1} - \mathbf{1})$
        $b_i(k+1) = b_i(k) + \rho(W_i(k+1)^T\mathbf{1} - \mathbf{1})$
        $M_i(k+1) = M_i(k) + \frac{\rho}{2}\sum_{j\in N_i}(W_i(k+1) - W_j(k+1))$
end for
```

Remark 2. (Convergence of the primal variable sequences) ADMM does not guarantee convergence of the primal variables [15, Section 3.2.1]. In the context of Theorem 1, this implies that, in general, $\lim_{k\to\infty} W_i(k)$ may not exist. However, this does not affect the practical application of our result since we are only interested in a weight matrix W which satisfies the constraints of (P2) and gives an optimal convergence factor.

Remark 3. (Stopping criterion for Algorithm 1) A stopping criterion for Algorithm 1 is derived in Appendix B. The criterion can be verified locally as follows. Each agent fixes an arbitrary, small number $\epsilon > 0$. At each iteration k, the agent computes its 'residual' $R_i(k)$, which captures how far is the agent's estimate $W_i(k)$ from the optimal objective function value (see (26) further ahead for the definition of $R_i(k)$). The agent is trying to minimize this residual. For a given iteration k, we say the stopping criterion of agent i is satisfied if $R_i(k) \leq \epsilon$. Note that for the smallest k for which the stopping criterion of agent i is satisfied, the agent can stop updating its variables and send the fixed value $W_i = W_i(k)$ to its neighbours as long as $R_i(k) \leq \epsilon$ holds. We say that the stopping criterion for Algorithm 1 is satisfied if $R_i(k) \leq \epsilon$ for all $i \in \{1,\dots,n\}$.

Remark 4. (Running Algorithm 1 in a distributed, parallel manner) We can verify that Algorithm 1 can indeed be executed in a distributed manner. Each agent i maintains a primal variable $W_i(k)$ and dual variables $a_i(k)$, $b_i(k)$ and $M_i(k)$ at each iteration k. The penalty parameter $\rho > 0$ can be arbitrarily fixed by the agent, although a standard choice is $\rho = 1$. At each iteration, each agent updates its primal variable, exchanges the updated primal variables with its neighbours and updates its dual variables. Thus, Algorithm 1 can run in a distributed manner, where each agent has to exchange values only with its immediate neighbours.
From the primal update equation (13) of the algorithm, it is clear that to calculate $W_i(k+1)$, agent i requires the $W_j(k)$

values only from the previous iteration of its neighbours. Hence, it does not have to wait for other agents to finish their updates before performing its update. Thus, Algorithm 1 can run in parallel across all agents.

Remark 5. (Information required in running Algorithm 1) To run Algorithm 1 on all agents, the agents have to initialize their set of primal and dual variables with the same dimensions across all agents. This requires that each agent must know the total number of agents n. Since this may not be possible, the agents can have a common upper-estimate of n and initialize their variables as per this estimate.
Each agent i also has to know *which agents* are its neighbours to incorporate the constraint $(W_i)_{ij} = 0, (i,j) \notin E$ in its primal update step (13). For this, all the agents should be indexed a priori and each agent must know its index. Then, in the first iteration of the algorithm, each agent can communicate with its neighbours to know their indices.

Next, we present some numerical examples to illustrate the convergence of Algorithm 1 and to compare our method with the centrally computed W^* and the Metropolis weight matrix W_{M}.

IV. Examples

A. *Fixed network*

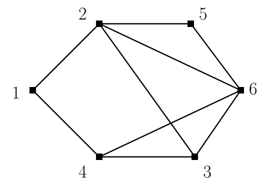

Fig. 1. A connected network of $n = 6$ agents

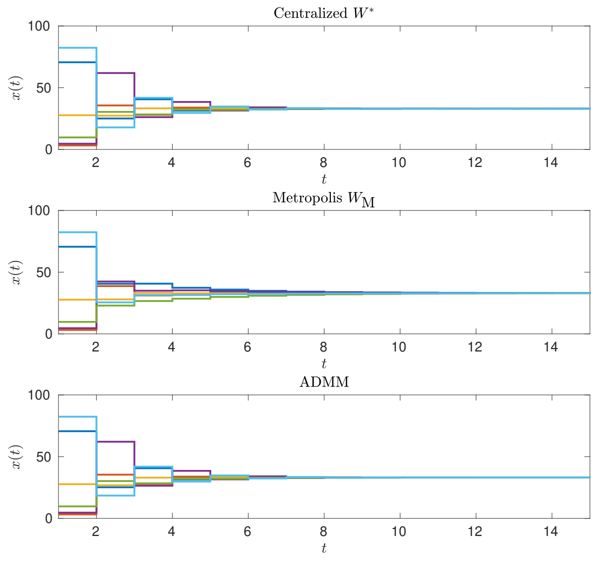

Fig. 2. Convergence of the agent values to the average of the initial values with the three weight matrices W^*, W_{M} and W_{admm} used in the consensus protocol for the network shown in Fig. 1

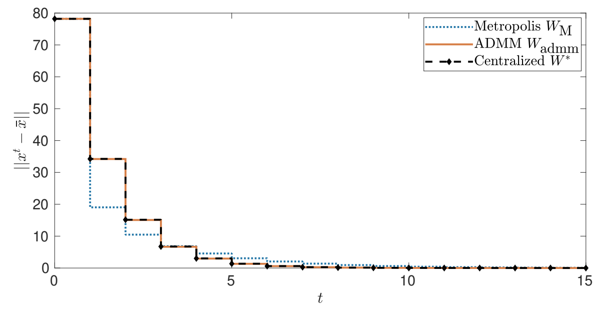

Fig. 3. Convergence to zero of the consensus error with the three weight matrices W^*, W_{M} and W_{admm} used in the consensus protocol for the network shown in Fig. 1. It was observed that the error went below 0.1 at $t = 9$ for the centralized and ADMM methods, while the Metropolis weights required $t = 14$.

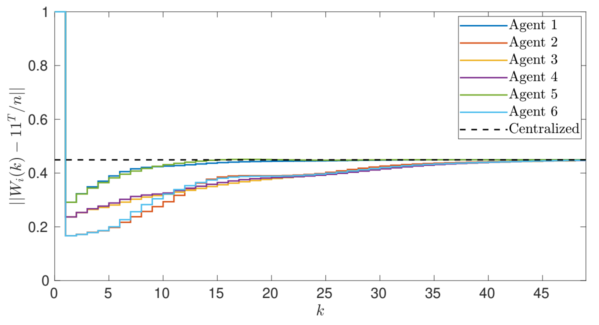

Fig. 4. Convergence of the objective functions maintained by the agents to the centralized optimal value $||W^* - \mathbf{1}\mathbf{1}^T/n||$ as Algorithm 1 progresses for the network shown in Fig. 1

We consider an undirected, connected network of $n = 6$ agents as shown in Fig. 1. For this network, we solve[1] the centralized problem (P2) to get the optimal weight matrix W^*. Then, we compute the Metropolis weight matrix W_{M} using (8).

Now, to calculate the weight matrix using our method, we run Algorithm 1 in parallel on all 6 agents using $\rho = 1/16$ as the penalty parameter[2], $\epsilon = 0.001$ for the stopping criteria. It was observed that the stopping criterion of Algorithm 1 was satisfied at $k = 48$ (refer Remark 3 for details on how to check the stopping criterion). We want to find the convergence factor of the consensus protocol (2) obtained using our method. For this, we define

$$W_{\mathrm{admm}} = \begin{bmatrix} w_1^T \\ \vdots \\ w_n^T \end{bmatrix},$$

where $w_i \in \mathbb{R}^{n\times 1}$ is the i^{th} row of $W_i(48)$. The convergence factors of W_{admm} and other weight matrices are shown in Table I. It can be seen that the convergence factor of W_{admm} is quite close to that of the centrally computed W^* and is much better than that of W_{M}. Now, we run the consensus protocol using these different weight matrices. Consider the

[1]We solve all optimization problems using the MOSEK solver [20] in YALMIP toolbox [21] on MATLAB [22].

[2]This value of ρ was observed to give faster convergence of Algorithm 1 for our examples.

	Centrally computed	Locally computed	
	W^* (centralized)	W_{M} (Metropolis)	W_{admm} (Algorithm 1)
$\|\|W - \mathbf{1}\mathbf{1}^T/n\|\|$	0.4492	0.6724	0.4519

TABLE I
CONVERGENCE FACTORS OF DIFFERENT WEIGHT MATRICES FOR THE NETWORK SHOWN IN FIG. 1

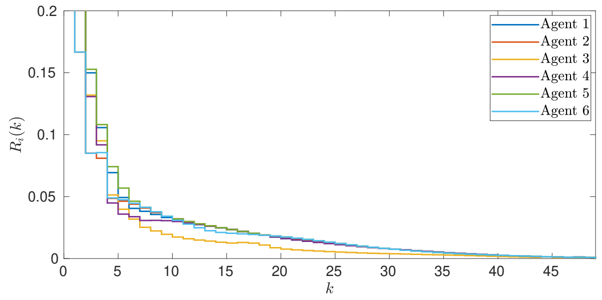

Fig. 5. Convergence to zero of the maximum residuals maintained by the agents as Algorithm 1 progresses for the network shown in Fig. 1. The residuals become smaller than $\epsilon = 0.001$ (i.e., the stopping criterion is satisfied) at $k = 48$.

randomly generated initial agent values given by $x(0) = \begin{bmatrix}70.6046 & 3.1833 & 27.6923 & 4.6171 & 9.7132 & 82.3458\end{bmatrix}^T$. Their average is $x_{\text{avg}}(0) = 33.0260$. Fig. 2 shows that the agents reach consensus to this average value using each of the three weight matrices. Fig. 3 shows that for the centralized and ADMM methods, the consensus error $e(t) = x(t) - \bar{x}$ decays at almost the same rate, while the decay is slower for the Metropolis weights.

Next, we illustrate the convergence of Algorithm 1 used in calculating W_{admm}. Fig. 4 shows the convergence of the objective functions maintained by each agent to the centralized optimal value $||W^* - \mathbf{1}\mathbf{1}^T/n|| = 0.4492$. This illustrates Statement 3 of Theorem 1. Fig. 5 shows the convergence to zero of the maximum residual at each agent as defined in (26). Intuitively, the maximum residual $R_i(k)$ of agent i captures how far is the objective function value $||W_i(k) - \mathbf{1}\mathbf{1}^T/n||$ maintained by the agent from the optimal objective value $||W^* - \mathbf{1}\mathbf{1}^T/n||$ (refer Appendix B for details on the maximum residual). Comparing Fig. 5 with Fig. 4, we can observe that the objective functions are close to the optimal value when the residuals are close to zero. This asserts that our choice of the stopping criterion as derived in Appendix B is a good one.

B. New agents enter a network: ADMM live — a variation of Algorithm 1

In the previous example, we compared the convergence factor of W_{admm} with the centralized W^* and the Metropolis W_{M} and observed that the performance of W_{admm} is better than W_{M}. However, Algorithm 1 requires some iterations of communication and computation in finding W_{admm} in a distributed manner. While certain applications do have some initial buffer time to compute the weights, few applications might not have this. For such cases we propose the following variation where at each iteration of Algorithm 1, we employ the consensus protocol (2) using the *latest available* weights. More specifically, for each iteration $k \geq 0$, define

$$\hat{W}_{\text{admm}}(k) = \begin{bmatrix} w_1(k)^T \\ \vdots \\ w_n(k)^T \end{bmatrix},$$

where $w_i(k) \in \mathbb{R}^{n\times 1}$ is the i^{th} row of $W_i(k)$. Note that for small values of k, the matrix $\hat{W}_{\text{admm}}(k)$ may not satisfy the constraint

$$\hat{W}_{\text{admm}}(k)\mathbf{1} = \mathbf{1},\ \hat{W}_{\text{admm}}(k)^T\mathbf{1} = \mathbf{1}. \tag{9}$$

However, it is necessary that a weight matrix satisfies this constraint at each instant of time for the agents to achieve average consensus using the consensus protocol (refer Proposition 1).

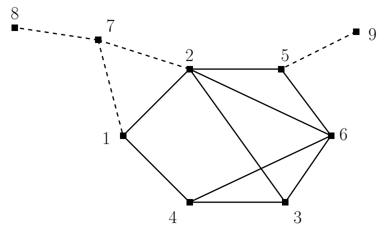

Fig. 6. At $t = 0$, agents 1 to 6 are connected by a network. At $t = 30$, agents 7, 8 and 9 enter the network.

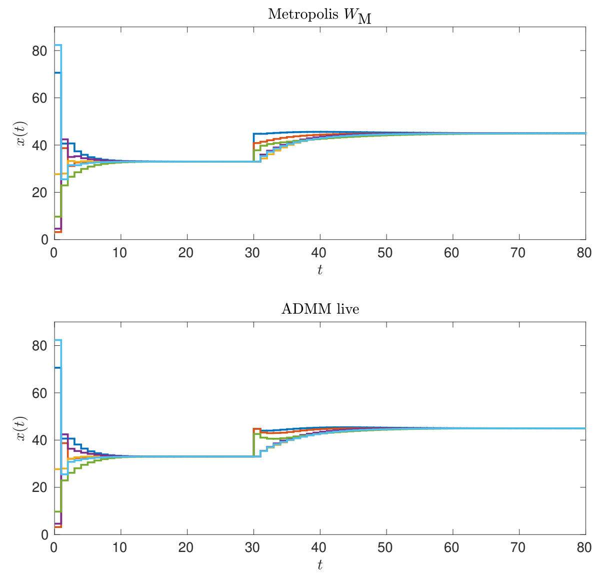

Fig. 7. Evolution of the agent values with W_{M} and ADMM live for the network shown in Fig. 6. In both methods, the agents reach consensus to the new average value after new agents enter the network at $t = 30$.

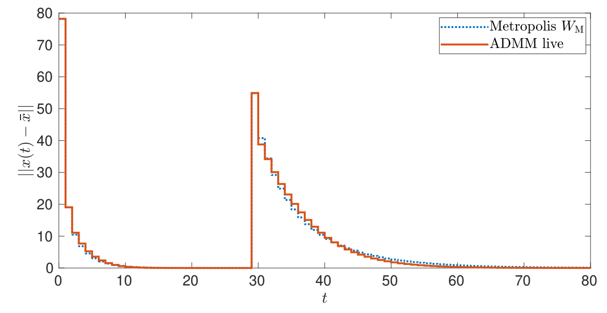

Fig. 8. Evolution of the consensus error with W_{M} and ADMM live for the network shown in Fig. 6. It was observed that, after the new agents entered the network at $t = 30$, the error went below 0.1 at $t = 67$ for ADMM live, while the Metropolis weights required $t = 80$.

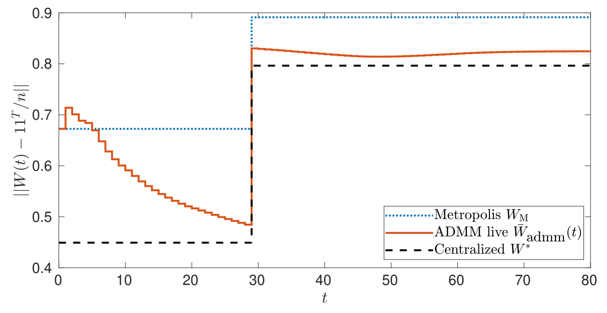

Fig. 9. Convergence factor of the consensus protocol with W_{M} and ADMM live for the network shown in Fig. 6

Hence, we modify $\hat{W}_{\text{admm}}(k)$ as follows. Define a new weight matrix $\bar{W}_{\text{admm}}(k)$ as

$$\bar{W}_{\text{admm}}(k)_{ij} = \begin{cases} \min\{\hat{W}_{\text{admm}}(k)_{ij}, \hat{W}_{\text{admm}}(k)_{ji}\}, & i \neq j, \\ 1 - \sum_{l \neq i, l=1}^{n} \bar{W}_{\text{admm}}(k)_{il}, & i = j, \end{cases}$$

i.e., at each iteration k, we convert $\hat{W}_{\text{admm}}(k)$ into a symmetric matrix by replacing its $(i,j)^{\text{th}}$ and $(j,i)^{\text{th}}$ elements with the smaller one among the two. In context of Algorithm 1, this can be done by exchanging weights among the neighbours. Then, each agent i adjusts its self-weight such that the row (and hence the column) sums of $\bar{W}_{\text{admm}}(k)$ are 1. This technique is same as the one used in calculating the Metropolis weight matrix W_{M} in (8). Now, $\bar{W}_{\text{admm}}(k)$ satisfies the constraint (9). Further, to reduce the overall time required to reach consensus, we initialize $W_i(k)$ using the Metropolis weights, i.e., for each $i \in \{1, \ldots, n\}$, set the i^{th} row of $W_i(0)$ to be the same as that of W_{M}, with the rest of the rows being zero. This way, the agent values in the consensus protocol will not remain 'idle' for $t = 0$ due to all weights being initialized at zero. Now, we update the agent values using $\bar{W}_{\text{admm}}(k)$ in the consensus protocol, i.e., for all $t \geq 0$,

$$x(t+1) = \bar{W}_{\text{admm}}(t)x(t). \tag{10}$$

We call the above method of implementing Algorithm 1 and consensus protocol (10) simultaneously as *ADMM live*. Now, we compare the performance of ADMM live with the Metropolis weights using the following example.

At time $t = 0$, suppose there are 6 agents connected by the same network as shown in Fig. 1. Now, at $t = 30$, three new agents enter the network as shown in Fig. 6. Suppose these new agents have 'initial values' $x_7(30) = 80.0559, x_8(30) = 74.5847, x_9(30) = 52.1186$. Now, the new average value of the 9 agents is $x_{\text{avg}}(0) = 44.9906$. Then, all the agents must compute a set of weights which involve these new agents and reach consensus to the new average value. The evolution of the agent values using the Metropolis W_{M} and ADMM live is shown in Fig. 7. Note that the centralized method is not capable of dynamically computing new weights after a change in the network topology. Hence, we compare ADMM live with the Metropolis weights, both of which can handle a change in network topology since they compute weights locally. In Fig. 7, the agents try to reach consensus initially, until new agents enter the network at $t = 30$. At this point, all agents compute a new set of weights and reach the new average value asymptotically. Fig. 8 shows the consensus error for the two methods. It is observed that since the convergence factor of the weights obtained by ADMM live converges to the optimal value, the error goes to zero faster than with the Metropolis W_{M}. In Fig. 9, we compare this changing convergence factor with the that of the fixed Metropolis weights. We can see that the convergence factor given by ADMM live is smaller than the Metropolis weights for most of the time. In fact, it can be observed from the figure that just after a few steps (at $k = 6$), the convergence factor of ADMM live starts outperforming that of the Metropolis weights. Also, this value approaches the optimal value given by the centralized optimal solution W^*. Hence we observe that the performance of ADMM live is better than that of Metropolis.

C. *Average computation time of Algorithm 1*

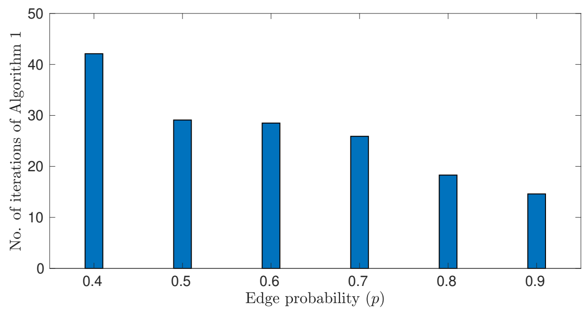

Fig. 10. Number of iterations required for Algorithm 1, averaged over 10 Erdos-Renyi random graphs. Each graph has $n = 10$ nodes and an edge probability of p.

We analyze what is the average number of iterations required for Algorithm 1 to satisfy its stopping criterion. For this, we construct a set of Erdos-Renyi (ER) random graphs as follows. Consider a network of $n = 10$ agents. For any pair of nodes, we place an edge between them with a probability p. For each fixed $p \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, we generate 10 ER random graphs. Then, we run Algorithm 1 on each of them with $\epsilon = 0.01$ for the stopping criterion. For each p, we take the average of the number of iterations required for the algorithm to stop, averaged over all 10 random graphs. The result is shown in Fig. 10. It can be seen that for denser graphs (larger p), the algorithm converges faster.

Next, we give some concluding remarks.

V. Conclusion

We proposed a method for agents in a network to locally compute a set of weights for the discrete-time distributed average consensus protocol. The weights are computed through an iterative algorithm derived from ADMM. The algorithm requires each agent to maintain and update a set of variables by solving an optimization problem and by exchanging these variables with its neighbours. We showed that the convergence factor of the consensus protocol with these locally computed weights converges to the optimal value given by the centralized solution. Through a numerical example, we showed that these weights perform better than the locally computed Metropolis weights in terms of the convergence factor.

Appendix A
Proof of Theorem 1

We have to show that Algorithm 1 is an ADMM implementation of problem (P3).

Remark 6. Note that if we try to solve problem (P3) directly using ADMM, although the ADMM algorithm can run in a distributed manner, the $W_i(k+1)$ updates will have to be performed 'serially' across different agents, i.e., agent i would have to wait for agents $1,\dots,i-1$ to update their values before it can perform its update. This is because the W_i's corresponding to different agents appear in the same constraint $W_i = W_j, (i,j) \in E$ in problem (P3).

To overcome the issue stated in Remark 6 and to enable 'parallel' implementation of ADMM, we introduce dummy variables[3] X_{ij}, X_{ji} for each $(i,j) \in E$ in (P3). Now, we get

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \frac{\|W_i - \mathbf{1}\mathbf{1}^T/n\|}{n} && \text{(P4)} \\ \text{subject to} \quad & W_i\mathbf{1} = \mathbf{1},\ W_i^T\mathbf{1} = \mathbf{1},\ i \in \{1,\dots,n\}, \\ & (W_i)_{ij} = 0, (i,j) \notin E, i \in \{1,\dots,n\}, \\ & W_i = X_{ij},\ W_j = X_{ji},\ X_{ij} = X_{ji},\ (i,j) \in E. \end{aligned}$$

Thus, the W_i's corresponding to different agents have now been 'decoupled'. Above problem is equivalent to (P3). To solve this problem using ADMM, we define the augmented Lagrangian for the problem as given in (11). Here, $\rho > 0$ is the penalty parameter and a_i, b_i, C_{ij}, D_{ij} are the dual variables for the respective constraints. Algorithm 2 is the standard ADMM algorithm for (P4). Note that the constraints $X_{ij} = X_{ji}, (i,j) \in E$ and $(W_i)_{ij} = 0, (i,j) \notin E$ have not been dualized but are incorporated in the primal update steps of the algorithm.

Algorithm 2 in its original form cannot be run in parallel. One of the reasons for this is that for any agent i, the $W_i(k+1)$ update is the argmin of L_ρ evaluated at $W_1 = W_1(k+1), \dots, W_{i-1} = W_{i-1}(k+1)$. Thus, agent i has to wait for the 'previous agents' to perform their updates and hence the issue mentioned in Remark 6 still exists. However, due to our introduction of X_{ij}, X_{ji}'s, with some algebraic manipulations, we can show that Algorithm 1 and Algorithm 2 are equivalent. The former can be executed in a parallel manner as argued in Remark 4.

Algorithm 2 ADMM implementation of problem (P3)

```
Initialize: $\rho > 0, W_i(0) = 0_{n\times n}, X_{ij}(0) = 0_{n\times n}, X_{ji}(0) = 0_{n\times n}, a_i(0) = 0_{n\times 1}, b_i(0) = 0_{n\times 1}, C_{ij}(0) = 0_{n\times n}, D_{ij}(0) = 0_{n\times n}$
Iterate:
for $k \geq 0$ do
    Primal update:
        $W_i(k+1) = \underset{W_i:(W_i)_{ij}=0,(i,j)\notin E}{\operatorname{argmin}} L_\rho, i \in \{1,\dots,n\}$
        $X_{ij}(k+1) = \underset{X_{ij}:X_{ij}=X_{ji}}{\operatorname{argmin}} L_\rho, (i,j) \in E$
        $X_{ji}(k+1) = \underset{X_{ji}:X_{ij}=X_{ji}}{\operatorname{argmin}} L_\rho, (i,j) \in E$
    Dual update:
        $a_i(k+1) = a_i(k) + \rho(W_i(k+1)\mathbf{1} - \mathbf{1}), i \in \{1,\dots,n\}$
        $b_i(k+1) = b_i(k) + \rho(W_i(k+1)^T\mathbf{1} - \mathbf{1}), i \in \{1,\dots,n\}$
        $C_{ij}(k+1) = C_{ij}(k) + \rho(W_i(k+1) - X_{ij}(k+1)), (i,j) \in E$    (14)
        $D_{ij}(k+1) = D_{ij}(k) + \rho(W_j(k+1) - X_{ji}(k+1)), (i,j) \in E$    (15)
end for
```

Lemma 2. *Algorithm 1 is equivalent to Algorithm 2.*

Proof. We will simplify the primal and dual update steps in Algorithm 2 to arrive at Algorithm 1.

Consider an arbitrary $(i,j) \in E$. The primal update $X_{ij}(k+1)$ can be evaluated by equating the gradient of L_ρ with respect to X_{ij} to zero. We enforce the constraint $X_{ij} = X_{ji}$ when evaluating the gradient. This gives the following equation.

$$\begin{aligned} -C_{ij}(k) - D_{ij}(k) + \rho[X_{ij}(k+1) - W_i(k+1) \\ + X_{ij}(k+1) - W_j(k+1)] = 0 \end{aligned} \tag{16}$$

This implies

$$\begin{aligned} X_{ij}(k+1) &= X_{ji}(k+1) \\ &= \frac{C_{ij}(k) + D_{ij}(k)}{2\rho} + \frac{W_i(k+1) + W_j(k+1)}{2}. \end{aligned} \tag{17}$$

[3]This trick is taken from [23, Section 3.4].

Augmented Lagrangian:

$$L_\rho = \frac{1}{n}\sum_{i=1}^{n} ||W_i - \mathbf{1}\mathbf{1}^T/n|| + \sum_{i=1}^{n} a_i^T(W_i\mathbf{1}-\mathbf{1}) + \sum_{i=1}^{n} b_i^T(W_i^T\mathbf{1}-\mathbf{1}) + \sum_{(i,j)\in E}\Big[\mathrm{Tr}((W_i - X_{ij})^T C_{ij}) + \mathrm{Tr}((W_j - X_{ji})^T D_{ij})\Big] + \frac{\rho}{2}\Big[\sum_{i=1}^{n}\big(||W_i\mathbf{1}-\mathbf{1}||^2 + ||W_i^T\mathbf{1}-\mathbf{1}||^2\big) + \sum_{(i,j)\in E}\big(||W_i - X_{ij}||_F^2 + ||W_j - X_{ji}||_F^2\big)\Big] \tag{11}$$

Update rule in Algorithm 2:

$$W_i(k+1) = \operatorname*{argmin}_{W_i:(W_i)_{ij}=0,(i,j)\notin E}\Big\{\frac{1}{n}||W_i - \mathbf{1}\mathbf{1}^T/n|| + a_i(k)^T(W_i\mathbf{1}-\mathbf{1}) + b_i(k)^T(W_i^T\mathbf{1}-\mathbf{1}) + \sum_{j\in N_i}\mathrm{Tr}(W_i^T C_{ij}(k)) + \frac{\rho}{2}\Big[||W_i\mathbf{1}-\mathbf{1}||^2 + ||W_i^T\mathbf{1}-\mathbf{1}||^2 + \sum_{j\in N_i}||W_i - X_{ij}(k)||_F^2\Big]\Big\} \tag{12}$$

Update rule in Algorithm 1:

$$W_i(k+1) = \operatorname*{argmin}_{W_i:(W_i)_{ij}=0,(i,j)\notin E}\Big\{\frac{1}{n}||W_i - \mathbf{1}\mathbf{1}^T/n|| + a_i(k)^T(W_i\mathbf{1}-\mathbf{1}) + b_i(k)^T(W_i^T\mathbf{1}-\mathbf{1}) + \mathrm{Tr}(W_i^T M_i(k)) + \frac{\rho}{2}\Big[||W_i\mathbf{1}-\mathbf{1}||^2 + ||W_i^T\mathbf{1}-\mathbf{1}||^2 + \sum_{j\in N_i}\Big|\Big|W_i - \frac{W_i(k)+W_j(k)}{2}\Big|\Big|_F^2\Big]\Big\}, \tag{13}$$

where $M_i(k) = \sum_{j\in N_i} C_{ij}(k)$.

Substituting these in the dual updates (14), (15), we have

$$C_{ij}(k+1) = C_{ij}(k) + \rho\Big(\frac{W_i(k+1)-W_j(k+1)}{2} - \frac{C_{ij}(k)+D_{ij}(k)}{2\rho}\Big), \tag{18}$$

$$D_{ij}(k+1) = D_{ij}(k) + \rho\Big(\frac{W_j(k+1)-W_i(k+1)}{2} - \frac{C_{ij}(k)+D_{ij}(k)}{2\rho}\Big). \tag{19}$$

Adding the above two equations, we get

$$C_{ij}(k+1) + D_{ij}(k+1) = 0.$$

Initializing $C_{ij}(0) = D_{ij}(0) = 0$ implies

$$C_{ij}(k) + D_{ij}(k) = 0 \tag{20}$$

for all $k \geq 0$. Substituting this back in (18), (19) gives

$$C_{ij}(k+1) = C_{ij}(k) + \rho\Big(\frac{W_i(k+1)-W_j(k+1)}{2}\Big), \tag{21}$$

$$D_{ij}(k+1) = D_{ij}(k) + \rho\Big(\frac{W_j(k+1)-W_i(k+1)}{2}\Big). \tag{22}$$

Further, substituting $C_{ij}(k) + D_{ij}(k) = 0$ in (17) gives

$$X_{ij}(k+1) = X_{ji}(k+1) = \frac{W_i(k+1)+W_j(k+1)}{2}. \tag{23}$$

Next, consider an arbitrary $i \in \{1,\ldots,n\}$. Then, the update $W_i(k+1)$ in Algorithm 2 can be evaluated as given in (12). Here, in finding the argmin of L_ρ, we have ignored those terms in L_ρ which are independent of W_i.

Now, in (12), we can substitute $X_{ij}(k)$ from (23). Further, we can simplify the term $\sum_{j\in N_i}\mathrm{Tr}(W_i^T C_{ij}(k))$ as follows. Define $M_i(k) = \sum_{j\in N_i} C_{ij}(k)$. Then,

$$\begin{aligned} M_i(k+1) &= \sum_{j\in N_i} C_{ij}(k+1) \\ &= \sum_{j\in N_i}\Big[C_{ij}(k) + \rho\Big(\frac{W_i(k+1)-W_j(k+1)}{2}\Big)\Big] \\ &= M_i(k) + \rho\sum_{j\in N_i}\Big(\frac{W_i(k+1)-W_j(k+1)}{2}\Big) \end{aligned}$$

and $\sum_{j\in N_i}\mathrm{Tr}(W_i^T C_{ij}(k)) = \mathrm{Tr}(W_i^T M_i(k))$. Thus, $W_i(k+1)$ can be computed as given in (13).

In conclusion, going from Algorithm 2 to Algorithm 1, the primal variables $X_{ij}(k), X_{ji}(k)$ are no longer required to be maintained explicitly. Further, the dual variables $C_{ij}(k), D_{ij}(k)$ have been combined into $M_i(k)$. The other dual variables $a_i(k), b_i(k)$ remain unchanged.

□

Appendix B
Stopping criterion for Algorithm 1

We derive a stopping criterion for Algorithm 1. A good stopping criterion for the ADMM of a constrained optimization problem is when the optimality conditions of the problem are satisfied with some tolerance [15]. The optimality conditions for problem (P4) are that the gradient of the *unaugmented* Lagrangian L_0 ($\rho = 0$ in (11)) with respect to the primal variables W_i and X_{ij} must be zero and the constraints must be satisfied, i.e., for all $i \in \{1, \ldots, n\}$,

$$\begin{aligned} &\frac{\partial}{\partial W_i} L_0 = 0,\ \frac{\partial}{\partial X_{ij}} L_0 = 0, j \in N_i,\\ &W_i \mathbf{1} = \mathbf{1},\ W_i^T \mathbf{1} = \mathbf{1},\\ &(W_i)_{ij} = 0, j \notin N_i,\ W_i = W_j, j \in N_i. \end{aligned} \tag{24}$$

We check under what conditions do the primal and dual variables in Algorithm 2 satisfy these conditions.

First, we show that the conditions $(\partial/\partial W_i)L_0 = 0$ and $(\partial/\partial X_{ij})L_0 = 0$ are always satisfied by the primal and dual variable iterates $W_i(k+1), a_i(k+1), b_i(k+1), C_{ij}(k+1), D_{ij}(k+1)$ for all $k \geq 0$.

By definition of $W_i(k+1)$ as given in Algorithm 2, we know that $W_i(k+1)$ minimizes L_ρ given in (11). Hence, $(\partial/\partial W_i)L_\rho$ evaluated at $W_i = W_i(k+1)$ must be zero. This means

$$\begin{aligned} &\frac{\partial}{\partial W_i}\left(\frac{1}{n}||W_i - \mathbf{1}\mathbf{1}^T/n||\right)\Bigg|_{W_i = W_i(k+1)} + a_i(k)\mathbf{1}^T\\ &+ \mathbf{1}b_i(k)^T + \sum_{j \in N_i} C_{ij}(k) + \rho\Bigg[(W_i(k+1)\mathbf{1} - \mathbf{1})\mathbf{1}^T\\ &+ \mathbf{1}(W_i(k+1)^T\mathbf{1} - \mathbf{1}) + \sum_{j \in N_i}(W_i(k+1) - X_{ij}(k))\Bigg] = 0. \end{aligned}$$

From the dual update equations $a_i(k+1), b_i(k+1), C_{ij}(k+1)$ as given in Algorithm 2, we can combine some terms in the above equation to get

$$\begin{aligned} &\frac{\partial}{\partial W_i}\left(\frac{1}{n}||W_i - \mathbf{1}\mathbf{1}^T/n||\right)\Bigg|_{W_i = W_i(k+1)} + a_i(k+1)\mathbf{1}^T\\ &+ \mathbf{1}b_i(k+1)^T + \sum_{j \in N_i} C_{ij}(k+1) = 0. \end{aligned}$$

Above is precisely the condition $(\partial/\partial W_i)L_0 = 0$ evaluated at $W_i(k+1), a_i(k+1), b_i(k+1), C_{ij}(k+1)$. Thus, $(\partial/\partial W_i)L_0 = 0$ is always satisfied by the primal and dual variable iterates.

Now, consider the second condition $(\partial/\partial X_{ij})L_0 = 0$. Evaluating the gradient of L_0, with the constraint that $X_{ij} = X_{ji}$, this condition can be written as $C_{ij} + D_{ij} = 0$. From (20), we know that this is true for $C_{ij}(k), D_{ij}(k)$ for all $k \geq 0$. Thus, $(\partial/\partial X_{ij})L_0 = 0$ is always satisfied by the primal and dual iterates of the algorithm.

Thus, from the optimality conditions (24), only the constraint conditions are *not* satisfied by the iterates of the variables in Algorithm 2. Define the residuals of the constraints as

$$\begin{aligned} &r_i^1(k) = ||W_i(k)\mathbf{1} - \mathbf{1}||/\sqrt{n},\\ &r_i^2(k) = ||W_i(k)^T\mathbf{1} - \mathbf{1}||/\sqrt{n},\\ &r_{ij}^3(k) = ||W_i(k) - W_j(k)||_F/n, j \in N_i,\\ &r_{ij}^4(k) = |(W_i(k))_{ij}|, j \notin N_i, \end{aligned} \tag{25}$$

for all $i \in \{1, \ldots, n\}$. Here, we have divided by n or $\sqrt{n}$ to compensate for the dimension of the quantity inside the norm. Now, we define the maximum residual at agent i as

$$R_i(k) = \max\{r_i^1(k), r_i^2(k), r_{ij}^3(k), r_{ij}^4(k) : j \in N_i\}. \tag{26}$$

Then, the optimality conditions (24) are satisfied with an ϵ tolerance for some k if

$$R_i(k) \leq \epsilon \text{ for all } i \in \{1, \ldots, n\}. \tag{27}$$

This is the stopping criterion for Algorithm 1. Remark 3 explains how this stopping criterion is used in the algorithm.

Appendix C
Proof of Lemma 1

First, we recall the definition of a primitive matrix.

Definition 1. A non-negative matrix W is said to be primitive with index $m \geq 1$ if $W^m > 0$.

Primitive matrices are a special class of matrices which have all but one eigenvalue strictly within the circle described by the spectral radius of the matrix. We make use of this fact whose proof can be found in [24, (8.3.16)].

Proposition 2. *[24] Suppose W is a non-negative matrix which satisfies $W\mathbf{1} = \mathbf{1}, W^T\mathbf{1} = \mathbf{1}$. Then, W is primitive if and only if $\rho(W - \mathbf{1}\mathbf{1}^T/n) < 1$.*

Given that the graph G is connected, using the above result, we now construct a weight matrix W which satisfies the consensus condition (4). We will use this result to prove Lemma 1.

Lemma 3. *Given G is a connected graph, consider a non-negative weight matrix W which satisfies*

$$W\mathbf{1} = \mathbf{1},\ W^T\mathbf{1} = \mathbf{1},\ W_{ij} > 0, (i,j) \in E. \tag{28}$$

Then, $\rho(W - \mathbf{1}\mathbf{1}^T/n) < 1$.

Proof. We prove this result by showing that W is a primitive matrix, i.e., we show that $\exists m \geq 1$ such that $W^m > 0$.

For a given pair of nodes (i,j), consider any $m_{ij} \geq 2$. Then,

$$(W^{m_{ij}})_{ij} = \sum_{l_1=1}^{n}\sum_{l_2=1}^{n}\cdots\sum_{l_{m_{ij}-1}=1}^{n} W_{il_1}W_{l_1l_2}\ldots W_{l_{m_{ij}-1}j}.$$

Note that for $m_{ij} = 1$, $(W^{m_{ij}})_{ij} = W_{ij}$. Thus, $(W^{m_{ij}})_{ij}$ is the sum of the product of weights of all edges which are part of all paths of length at most m_{ij} between nodes (i,j) of the graph. Since the graph G is connected, there is a path (of length at most $n-1$) between every pair of nodes (i,j), i.e., for all $(i,j) \in \{1, \ldots, n\}^2, \exists m_{ij} \in \{1, \ldots, n-1\}$ such that $(W^{m_{ij}})_{ij} > 0$. Hence, we can conclude that $W^{n-1} > 0$, i.e., W is primitive with an index of (at most) $n-1$.

Now, by Proposition 2, we have $\rho(W - \mathbf{1}\mathbf{1}^T/n) < 1$.

□

We are now ready to prove Lemma 1. We have to show that any solution W^* of (P2) satisfies $\rho(W^* - \mathbf{1}\mathbf{1}^T/n) < 1$. Since $\rho(A) \leq ||A||$ for any matrix $A \in \mathbb{R}^{n \times n}$, it is sufficient to show that $||W^* - \mathbf{1}\mathbf{1}^T/n|| < 1$. Moreover, since W^* is the optimal value of (P2), it is sufficient to show that, given G is connected, $\exists W \in \mathbb{R}^{n\times n}$ which satisfies the constraints

$$W\mathbf{1} = \mathbf{1},\ W^T\mathbf{1} = \mathbf{1},\ W_{ij} = 0, (i,j) \notin E$$

of (P2) such that $||W - \mathbf{1}\mathbf{1}^T/n|| < 1$. We construct such a W as follows. Consider a W which satisfies the condition (28). Then, we have

$$\begin{aligned}||W - \mathbf{1}\mathbf{1}^T/n||^2 &= \rho\big((W - \mathbf{1}\mathbf{1}^T/n)(W - \mathbf{1}\mathbf{1}^T/n)^T\big)\\ &= \rho(WW^T - \mathbf{1}\mathbf{1}^T/n).\end{aligned}$$

It is easy to verify that WW^T also satisfies the condition (28). This implies, $\rho(WW^T - \mathbf{1}\mathbf{1}^T/n) < 1$, which implies $||W - \mathbf{1}\mathbf{1}^T/n|| < 1$.

References

[1] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, April 2005, pp. 63–70.

[2] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," *Journal of Parallel and Distributed Computing*, vol. 7, no. 2, pp. 279 – 301, 1989.

[3] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, Sep. 2004.

[4] R. O. Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 2, June 2003, pp. 951–956.

[5] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[6] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65 – 78, 2004.

[7] Wei Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[8] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *2007 American Control Conference*, July 2007, pp. 711–716.

[9] L. Wang and F. Xiao, "Finite-time consensus problems for networks of dynamic agents," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 950–955, April 2010.

[10] C. Ko and L. Shi, "Scheduling for finite time consensus," in *2009 American Control Conference*, June 2009, pp. 1982–1986.

[11] S. Y. Shafi, M. Arcak, and L. El Ghaoui, "Designing node and edge weights of a graph to meet laplacian eigenvalue constraints," in *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep. 2010, pp. 1016–1023.

[12] Yoonsoo Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph laplacian," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 116–120, Jan 2006.

[13] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, vol. 46, no. 4, pp. 667–689, 2004.

[14] M. El Chamie, G. Neglia, and K. Avrachenkov, "Distributed weight selection in consensus protocols by schatten norm minimization," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1350–1355, May 2015.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, p. 1–122, Jan. 2011.

[16] J. Eckstein, "Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results," *RUTCOR Research Reports*, pp. 1–35, 2012.

[17] M. L. Overton and R. S. Womersley, "On minimizing the special radius of a nonsymmetric matrix function: Optimality conditions and duality theory," *SIAM Journal on Matrix Analysis and Applications*, vol. 9, no. 4, pp. 473–498, 1988.

[18] M. Mardani, G. Mateos, and G. B. Giannakis, "Decentralized sparsity-regularized rank minimization: Algorithms and applications," *IEEE Transactions on Signal Processing*, vol. 61, no. 21, pp. 5374–5388, Nov 2013.

[19] A. Zare, M. R. Jovanović, and T. T. Georgiou, "Alternating direction optimization algorithms for covariance completion problems," in *2015 American Control Conference (ACC)*, July 2015, pp. 515–520.

[20] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[21] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in matlab," in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[22] MATLAB, "version 9.2.0.556344 (r2017a)," in *The MathWorks Inc., Natick, Massachusetts, USA*, 2017.

[23] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. USA: Prentice-Hall, Inc., 1989.

[24] C. D. Meyer, *Matrix analysis and applied linear algebra*. SIAM, 2000, vol. 71.

Kiran Rokade received the B.Tech. degree in electrical engineering from V.J.T.I., Mumbai, India, in 2016. and the an M.S. degree in electrical engineering from Indian Institute of Technology Madras, Chennai, India in 2020. Currently, he is pursuing his PhD at Cornell University. His research interests include multi-agent systems, optimization theory and game theory.

Rachel Kalpana Kalaimani received the B.E. degree in electrical and electronics engineering from P.S.G. College of Technology, Coimbatore, India, in 2009 and the Ph.D. degree in control from the Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India, in 2014. After that she was a Post Doctoral Research Fellow at the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. She is currently an Assistant Professor at Indian Institute of Technology Madras, Chennai, India. Her current research interests include distributed optimization, networked control systems and complex systems. Other interests include optimization of energy consumption in buildings, model predictive control, graph theoretic techniques, and numerical linear algebra.