

A SYMMETRIC ADAPTIVE ALGORITHM FOR SPEEDING-UP CONSENSUS

Daniel Thai, Elizabeth Bodine-Baron, and Babak Hassibi

Department of Electrical Engineering
California Institute of Technology
thai, eabodine, hassibi@caltech.edu

ABSTRACT

Performing distributed consensus in a network has been an important research problem for several years, and is directly applicable to sensor networks, autonomous vehicle formation, etc. While there exists a wide variety of algorithms that can be proven to asymptotically reach consensus, in applications involving time-varying parameters and tracking, it is often crucial to reach consensus “as quickly as possible”. In [?] it has been shown that, with global knowledge of the network topology, it is possible to optimize the convergence time in distributed averaging algorithms via solving a semi-definite program (SDP) to obtain the optimal averaging weights. Unfortunately, in most applications, nodes do not have knowledge of the full network topology and cannot implement the required SDP in a distributed fashion. In this paper, we present a symmetric adaptive weight algorithm for distributed consensus averaging on bi-directional noiseless networks. The algorithm uses an LMS (Least Mean Squares) approach to adaptively update the edge weights used to calculate each node’s values. The derivation shows that global error can be minimized in a distributed fashion and that the resulting adaptive weights are symmetric—symmetry being critical for convergence to the true average. Simulations show that convergence time is nearly equal to that of a non-symmetric adaptive algorithm developed in [?], and significantly better than that of the non-adaptive Metropolis-Hastings algorithm. Most importantly, our symmetric adaptive algorithm converges to the sample mean, whereas the method of [?] converges to an arbitrary value and results in significant error.

Index Terms— Adaptive Consensus, LMS algorithm, Sensor Network

1. INTRODUCTION

Consensus is an important problem and has received much attention in the literature on sensor networks and distributed algorithms (see e.g., [?, ?, ?, ?, ?, ?] and the references

This work was supported in parts by the National Science Foundation under grant CCF 0729203, by the Office of Naval Research under grant N00014-08-1-0747, by the David and Lucille Packard Foundation, and by Caltech’s Lee Center for Advanced Networking.

therein). A basic sensor network is made up of n nodes, each one of which is separated by a certain distance, and takes a measurement of some value. We assume that each sensor’s reading is independently corrupted by noise. It is often costly to transmit all n readings of the sensor network to the user, and in many cases, the readings must be aggregated into a single value. Therefore, we wish to compute the average value of all of the sensors’ readings for transmission to a base station.

Several methods exist to perform this task. One of the simplest is to designate a ‘super node.’ In this method, all other nodes transmit all of the values they have recorded to the ‘super node.’ [?]. The ‘super node’ receives all of the information, performs the averaging, and transmits to the user this single value. (This assumes that the transmitter is based at the ‘super node’.)

However, there are several problems with this method. The first is that the system can be easily rendered inoperative by destroying the ‘super node’. A second is that the system cannot easily react to additions and deletions of nodes; the ‘super node’ needs to know how many nodes there are. To solve the problems associated with a central controller, several papers have proposed various methods for performing distributed averaging [?, ?, ?, ?, ?, ?].

While there exists a wide variety of algorithms that can be proven to asymptotically reach consensus, in many applications (especially those involving time-varying parameters) time-to-convergence is also important. For the distributed averaging problem, minimizing the convergence time reduces to minimizing the second largest eigenvalue of the weight matrix used by the network to perform consensus. When the network topology is known to all nodes in the network, this minimization can be done via solving a semi-definite program (SDP) [?]. Unfortunately, in most applications, this is an unrealistic assumption and distributed methods, which rely only on a node’s local knowledge of the network, need to be developed. In [?] an adaptive weight update method was proposed and shown to have faster convergence time than fixed-weight methods. Unfortunately, since the method of [?] results in non-symmetric weights it generally reaches consensus to a value unequal to the desired sample mean. In this paper, we propose a *symmetric* adaptive weight algorithm for distributed consensus averaging. Simulations show that con-

vergence time is nearly equal to that of the non-symmetric adaptive algorithm of [?] (which does not converge to the true sample mean), and significantly better than that of the non-adaptive Metropolis-Hastings algorithm.

We provide a brief background and some prior work in sections 2 and 3. Our symmetric adaptive method is described in section 4. Section 5 presents some discussions and simulations, and the paper is concluded in section 6.

2. BACKGROUND

We define a graph as a set of vertices and edges $G = (N, E)$, with nodes numbered $1 \dots n$, and edges connecting the different nodes. By assumption, all edges are error-free and the graph is bidirectional; i.e., if there exists an edge from node i to node j , then there will also exist an edge from node j to node i . To perform distributed averaging, in every time step t , each node i transmits its numerical value $x_i(t)$ to its immediate set of neighbors, N_i .

The graph connectivity is encoded in an adjacency matrix, where

$$A_{ij} = \begin{cases} 1 & (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Note that in general, self loops are not assumed; $A_{ii} = 0$. Throughout this paper, we assume that graphs are fully connected; i.e., there exists a route from every node to every other node, traveling along the edge set E . We then define a weight matrix W , based on the adjacency matrix A .

In the weight matrix, each edge has a corresponding edge weight. If there is no edge between two nodes, this is recorded as an edge weight of zero. Note that a weight matrix will also include self-weights. Each node i computes its estimate of the “global” average at time $t + 1$, from its own and its neighbors values at time t according to

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

To make the above a “weighted average”, we will insist that each row of the matrix W sum up to one, i.e.,

$$\sum_j W_{ij} = W_{ii} + \sum_{j \in N_i} W_{ij} = 1, \quad (2)$$

in which case (1) can be rewritten as

$$x_i(t + 1) = x_i(t) + \sum_{j \in N_i} W_{ij}(x_j(t) - x_i(t)). \quad (3)$$

If we construct the vector $\mathbf{x}(t)$ from the entries $x_i(t)$, we clearly have $\mathbf{x}(t + 1) = W\mathbf{x}(t) = W^{t+1}\mathbf{x}(0)$.

Since the rows of W sum to one, it is a stochastic matrix and clearly it has the “all one” vector $\mathbf{1}$ as a right eigenvector,

$W\mathbf{1} = \mathbf{1}$. Further, if all other eigenvalues of W are strictly inside the unit circle, then it is well known that as $t \rightarrow \infty$,

$$W^t \rightarrow \frac{1}{n}\mathbf{1}\mathbf{v}^T, \quad (4)$$

where \mathbf{v}^T is the corresponding left eigenvector $\mathbf{v}^T W = \mathbf{v}^T$, such that $\mathbf{v}^T \mathbf{1} = 1$. Thus, under these conditions, distributed weighted averaging converges to

$$\mathbf{x}(t) \rightarrow \mathbf{1} \frac{\mathbf{v}^T \mathbf{x}(0)}{n}, \quad (5)$$

which implies consensus to the value $\frac{\mathbf{v}^T \mathbf{x}(0)}{n}$. For this to be consensus to the sample mean, we require that $\mathbf{v}^T = \mathbf{1}^T$, which means that $\mathbf{1}^T W = \mathbf{1}^T$, i.e., that W is doubly stochastic—both its row and column sums are unity.

The simplest way to guarantee double stochasticity (and the one that requires only local information) is to insist that W be symmetric, $W_{ij} = W_{ji}$, which is what we shall do as part of the symmetric adaptive method in Section 4.

3. PREVIOUS WORK

For purposes of comparison, we simulate a number of different methods for reaching consensus [1,3,4], and we briefly present them here. Firstly, the Metropolis-Hastings method [2] is defined to be:

$$W_{ij} = \begin{cases} \frac{1}{1 + \max(d_i, d_j)} & j \in N_i, i \neq j \\ 1 - \sum_{j \in N_i} W_{ij} & i = j \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where d_i and d_j are the degrees of nodes i and j . Note that the weight matrix requires only local information and is symmetric—thus converging to the true mean.

The second method is that of [?], which minimizes the convergence time by minimizing the second largest (in absolute value) eigenvalue of W via the semi-definite program: minimize s subject to:

$$sI \preceq W - \mathbf{1}\mathbf{1}^T \preceq sI \quad (7)$$

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

This method provides provides the optimal linear weights for a given network topology. The drawback is that solving the above SDP requires global information of the network which is usually not available.

Finally, the method of [?] is an adaptive algorithm, that attempts to find the optimal weights. In other words, it uses time-varying weights. It begins with the Metropolis weights already presented, but then updates these weights as:

$$\mathbf{W}_i(t + 1) = \mathbf{W}_i(t) + \mu_i e_i(t)(\mathbf{x}'_i(t) - x_i(t)\mathbf{1}) \quad (8)$$

where $\mathbf{W}_i(t)$ designates the nonzero entries of the i -th column of W , $\mathbf{x}'_i(t)$ is the vector of data from the neighbors of i and where $e_i(t)$ is defined to be the error

$$e_i(t) = x_{di}(t) - \mathbf{W}_i(t)^T (\mathbf{x}'_i(t) - x_i(t)\mathbf{1}) \quad (9)$$

and $x_{di}(t)$, the so-called desired signal, is simply the actual mean viewed from node i

$$x_{di}(t) = \frac{1}{|N_i| + 1} (x_i(t) + x'_i(t)\mathbf{1}) \quad (10)$$

This method does speed up convergence relative to the Metropolis-Hastings algorithm; however, since it does not guarantee the symmetry of W , it does not converge to the true mean.

4. A SYMMETRIC ADAPTIVE METHOD

In [?], the authors propose an adaptive method that minimizes the local error (??) for each node in the network. However, it would be desirable to minimize global error if possible; we would expect that this might cause network convergence to the sample mean to be both faster and more accurate. We assume that the system is initialized with Metropolis weights, defined in the previous section. We define the global error as $J = \sum_i E(e_i^2(t))$ and as with [?], take the gradient.

$$\begin{aligned} \nabla \sum_i E(e_i^2(t)) &= 2[E(e_i(t))(x_j(t) - x_i(t)) \\ &\quad + E(e_j(t))(x_i(t) - x_j(t))] \end{aligned} \quad (11)$$

This gradient is taken only with respect to W_{ij} , and we also assume that $W_{ij} = W_{ji}$. Making this assumption allows us to preserve the doubly stochastic nature of this matrix, which follows from the fact that $W = W^T$ and $\sum W_{ij} = 1$. Using the steepest descent method,

$$\begin{aligned} W_{ij}(t+1) &= W_{ij}(t) + \mu[E(e_i(t))(x_j(t) - x_i(t)) \\ &\quad + E(e_j(t))(x_i(t) - x_j(t))] \end{aligned} \quad (12)$$

Using the LMS method to approximate the expectation by the instantaneous value, this becomes

$$\begin{aligned} W_{ij}(t+1) &= W_{ij}(t) + \mu[e_i(t)(x_j(t) - x_i(t)) \\ &\quad + e_j(t)(x_i(t) - x_j(t))] \end{aligned} \quad (13)$$

The term μ above represents the speed that the nodes react to $e_i(t)$, and it is a fixed value for all nodes. If μ is too large, then the system could become unstable and not converge as weights oscillate rapidly. Alternatively, if μ is too small, then it could take a long time to reach consensus. While analyzing the effect of μ is highly nontrivial, as the system is nonlinear because the recursions for $x(\cdot)$ and $W(\cdot)$ are coupled, Lyapunov arguments can be used to prove stability for small enough μ —we omit the details for brevity.

5. DISCUSSION

The symmetric adaptive method requires only local information. Firstly, in order to initialize the network with symmetric weights, nodes must know their immediate neighbors' weights; we use Metropolis weights as our initial weights. In addition, our method requires each node to calculate and transmit its error value $e_i(t)$ to its neighbors. We make the simplifying assumption that the time to transmit this additional value is negligible compared to the total time in each iteration. We present simulation results below, which compares the symmetric adaptive method against the method of [?], which represents the best possible linear weights, as well as Metropolis weights followed by non-adaptive averaging, and the method of [?], which utilizes adaptive but non-symmetric averaging and Metropolis initial weights. These graphs represent the average of 10,000 simulations on the network, starting from different random initial values. Each node's random value was drawn from $U[0, 1]$. These simulations were run on the network example figure presented by [?]. This network has 8 nodes and 17 edges, with a minimum degree of 3 and a maximum degree of 6.

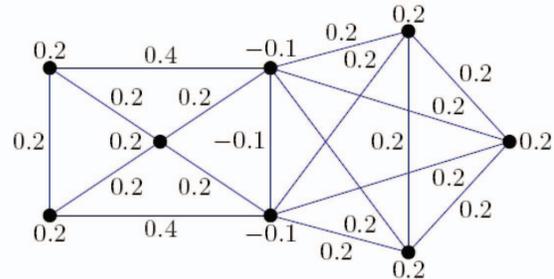


Fig. 1. This is the network upon which the simulations were run, and was the graph used by [?].

Stepsize used for the [?] method in simulation was

$$\mu_i = \frac{2}{(\sigma_x^2)|N_i|(|N_i| + 1)} \quad (14)$$

which is what [?] determined would be an upper bound. We also tried $\mu_i/10$ as suggested in [?], but this did not improve performance for this network. For our algorithm, we determined that the optimal stepsize μ was 1.31 for an initialization with Metropolis weights.

We note that the [?] method does not converge to the sample average (See figure 3). The Metropolis-Hastings method, while it does converge to the sample mean, is also significantly slower to converge than our method or that of [?]. However, it has the advantage of a simple implementation. Furthermore, the [?] method converges far faster to the sample mean than either of the previous methods but requires global information - the entire adjacency matrix. Worse yet, it must solve an SDP in order to determine the optimal weights

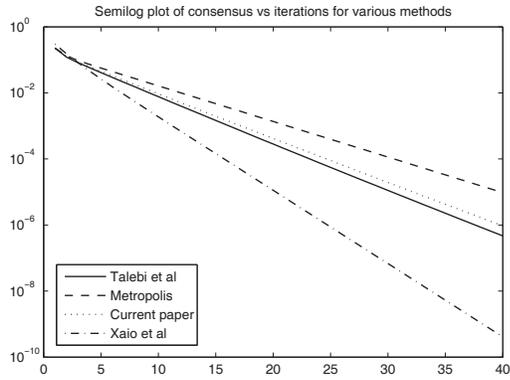


Fig. 2. This graph plots the difference between the maximum value of the network and the minimum value of the network.

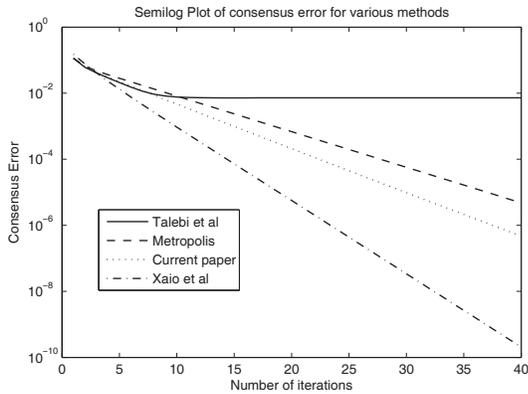


Fig. 3. This graph plots the difference between the maximum value of the network and the sample mean

[?]. This makes this method far less useful when dealing with networks where nodes may connect to and leave the network or where global information is inaccessible. Additionally, solving an SDP can be computationally complex and may require significant time to calculate weights before the consensus protocol begins. Our method, while neither as fast as [?], nor as simple as the Metropolis-Hastings method, fills a niche in between, allowing convergence to the mean faster than the Metropolis-Hastings method, while still requiring only local information. Its convergence time is similar to that of the method of [?], though it results in far less error, as seen in Figure 3, and as expected with the doubly stochastic matrix.

6. CONCLUSION

In the future, we hope to determine bounds on convergence time for our symmetric adaptive algorithm analytically, as well as a range of values for the learning rate μ . Clearly, we must also present a method of determining μ using only local

information. Other future directions include the addition of noise to the system, the possibility of time varying values or value tracking, link addition or deletion, and node addition or deletion, all of which add complexity, but also all of which are ideally suited to an adaptive system.