FINDING SPARSE CONNECTIVITY PATTERNS IN POWER-CONSTRAINED AD-HOC NETWORKS FOR ACCELERATING CONSENSUS ALGORITHMS

César Asensio-Marco, Baltasar Beferull-Lozano

Group of Information and Communication Systems
Instituto de Robótica y Tecnologías de la Información & las Comunicaciones (IRTIC)
Universidad de Valencia
46980, Paterna (Valencia), Spain
Email: {Cesar.Asensio, Baltasar.Beferull}@uv.es

ABSTRACT

In this paper, we show how to critically sparsify a given network while improving the convergence rate of the associated average consensus algorithm. Thus, instead of adding new links or reallocating them, we propose novel distributed methods to find much sparser networks with better convergence results than the original denser ones. We propose two distributed algorithms; a) in the first one, each node solves a local optimization problem using only its two-hop neighborhood, b) the second one is a distributed algorithm based on using, at each node, the power method. As compared with previous work, the reduction in the number of active links is doubled while improving the convergence rate and having a much lower power consumption. Simulation results are presented to verify and show clearly the efficiency of our approach.

1. INTRODUCTION

Two of the most important issues of the existing average consensus algorithms [1-6] are the convergence speed and the communication cost required to converge, which in most cases are the principal reasons for not using them in practical scenarios. Much of the research that focuses on improving these parameters has been carried out by Boyd [1][2]. They showed that it is possible to formulate the problem of finding the fastest converging iteration as a convex semidefinite optimization problem. This approach gives the optimal weights to combine the information based on a convex optimization problem. However, an important drawback of this method is that it requires knowing, at every node, the connectivity of the deployed network and secondly it requires too many computational resources for achieving the optimal solution. For solving the first mentioned problem, a subgradient algorithm was proposed in [1]. However this method is relatively slow, and has no simple stopping criterion that guarantees a certain level of suboptimality. Another solution is to set the neighboring link weights to a constant and optimize this constant to achieve maximum possible convergence rate. Since the global knowledge of the network spectrum is required to calculate this optimal constant, it is difficult to use this expression in a practical distributed consensus process.

This work was supported by the spanish MEC Grants TEC2009-14213 "SENSECOGNITIVE", CONSOLIDER-INGENIO 2010 CSD2008-00010 "COMONSENS" and the European STREP Project "SENDORA" Grant no. 216076 within the 7th European Community Framework Program.

For that reason, most of the existing related research [3] focuses on properly redesigning the original topology in order to improve the convergence time and power consumption of the associated average consensus algorithm, which in most cases, such as [4] and [5], involves adding several unrealistic large links to the original network. However, creating links between distant nodes might not be possible due to the communication power constraints that are present in battery supplied nodes. Another solution to improve the average consensus, by modifying the underlaying topology, consists on removing links instead of adding them. The authors of [6] show how to improve the convergence of the average consensus process by sparsifying the original network. It is shown that it is possible to achieve similar convergence results to the ones achieved using global knowledge of the original network as in [1] but using a sparsifyed version of the topology. The only information used in [6] is the local degree of nodes. This approach achieves quite good convergence results and reduces the power consumption by using a very simple and low complexity approach where no global knowledge is needed. In this work, we use the same general idea, but making use of a bite more of information while keeping low complexity and the possibility of distributed computations. We assume that each node knows a small part of the network, that is, its own neighborhood and the ones of its one hop neighbors (two-hop knowledge). Having this extra information, each node can locally create a small subnetwork from which extract useful graph spectrum information for removing suitable links. This leads to much sparser networks where the number of removed links is doubled in comparison with [6]. Moreover, these resulting networks present, at the same time, better convergence time, power consumption and the possibility of being obtained by only using distributed computations.

The rest of this paper is structured as follows: Section II presents some background on consensus problems. The motivation of our approach is given in Section III. In Section IV, we present a method for optimizing a splited version of the network. In Section V, we present a method for locating in a distributed way the suitable links, so that removing them, it is possible to achieve better convergence results. Section VI is devoted to validate our claims by comparing our results with existing approaches. Finally, conclusions are summarized in Section VII.

 ISSN 2076-1465

2. PROBLEM FORMULATION

A network of nodes can be modeled as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, consisting of a set $\mathbf{V}$ of N vertices and a set $\mathbf{E}$ of M edges. We denote an edge between vertices i and j as a pair (i, j), where the presence of an edge between two vertices indicates that they can communicate with each other.

Given a graph $\mathbf{G}$ we can assign an $N \times N$ adjacency matrix $\mathbf{A}$, given by:

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

The neighborhood of a node i is defined as

$$\Omega_i = \{j \in \mathbf{V} : (i,j) \in \mathbf{E}, \quad i = 1, ..., N\}$$

where $d_i = |\Omega_i|$ is the degree of node i.

Similarly, we denote by $\mathbf{L}$ the $N \times N$ Laplacian matrix of the graph, which is given by

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{1}$$

where $\mathbf{D} = \mathbf{diag}(d_1, ..., d_N)$ is the so-called degree matrix.

Let us assume that the sensor measurements of the nodes have some initial data at time slot $k = 0$. We collect them in a vector, which we call the initial state vector $\mathbf{x}(0)$, thus the average of the initial state $\mathbf{x}(0)$ is

$$\mathbf{x}_{avg} = \frac{\mathbf{1}\mathbf{1}^t \mathbf{x}(0)}{N}$$

where $\mathbf{1}$ denotes the all ones column vector. We consider the general linear update of the state of each sensor i at time k, using only local data exchange, namely

$$\mathbf{x}_i(k+1) = \sum_{j=1}^{N} W_{ij}(k)\mathbf{x}_j(k) \;\; \forall i = 1, 2..., N$$

where $\mathbf{W}$ denotes the mixing matrix, which in this paper, it is assumed to be given by:

$$\mathbf{W} = \mathbf{I} - \alpha\mathbf{L} \tag{2}$$

where α is a constant independent of time k, which we take as $\frac{1}{d_{max}}$ where d_{max} is the maximum degree in the network. Even though the optimal value of α, for a given connectivity graph is given by $\alpha = \frac{2}{\lambda_2(\mathbf{L}) + \lambda_N(\mathbf{L})}$, this choice requires to have a complete knowledge of the global connectivity graph, at every node, which is not scalable. On the other hand, the simple choice of $\alpha = \frac{1}{d_{max}}$ ensures that, at each time slot k, we give an equal weight to every available link and no global knowledge or processing is required for computing the weights. Since d_{max} is the maximum degree of the network it can be easily calculated in a distributed way, thus it is scalable.

The asymptotic convergence factor is defined as usual [1] by:

$$r(\mathbf{W}) = \sup \lim_{k \to \infty} \left(\frac{||\mathbf{x}(k) - \mathbf{x}_{avg}||}{||\mathbf{x}(0) - \mathbf{x}_{avg}||_2} \right)$$

which has been shown to be equal to:

$$r(\mathbf{W}) = \rho\left(\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{N}\right)$$

where $\rho(\mathbf{A})$ denotes the spectral radius of a matrix[1], and the associated convergence time denoted by $t(\mathbf{W})$ is given by:

$$t(\mathbf{W}) = \frac{-1}{\log(r(\mathbf{W}))} \tag{3}$$

The convergence time $t(\mathbf{W})$ is the main performance indicator we use in our work. In this paper, we show that it is possible to reduce the convergence time $t(\mathbf{W})$ by removing some properly chosen links, thus sparsifying a given graph. We also assume a communication cost associated with every link in the network. For the sake of simplicity, we assume in this paper that this cost is constant and proportional to the *squared distance* of the corresponding link. Then, the total communication cost $P(\mathbf{W})$ is proportional to $t(\mathbf{W})$ times the power consumption of one communication iteration across the various links of the associated gossip algorithm.

3. MOTIVATION OF OUR APPROACH

Our main goal in this work is to exploit the effect of improving the speed and power expenses of the average consensus by sparsifying a given dense network. The justification behind this idea is presented in detail in [6] and we revise it here for convenience. It has been shown in the literature [7] that given a graph $\mathbf{G}$, if we remove a link between two nodes of this graph, we get a new graph $\mathbf{G}'$ for which $\lambda_2(\mathbf{L}(\mathbf{G}')) \leq \lambda_2(\mathbf{L}(\mathbf{G}))$. Moreover, since the eigenvalues of $\mathbf{W}$ and $\mathbf{L}$ are related as follows:

$$\lambda_i(\mathbf{W}) = 1 - \alpha\lambda_i(\mathbf{L}) \tag{4}$$

if a link from the graph $\mathbf{G}$ is removed, we get that $\lambda_2(\mathbf{W}(\mathbf{G}')) \geq \lambda_2(\mathbf{W}(\mathbf{G}))$, which implies slower consensus convergence for the same value of α [1]. Note that by writing (4) in that way, we are assuming that the eigenvalues of the Laplacian matrix $\mathbf{L}$ are arranged in increasing order and the eigenvalues of the Weight matrix $\mathbf{W}$ are arranged in decreasing order. Then, if we take α as $\frac{1}{d_{max}}$ and we consider the action of removing a particular link, this gives rise to a reduction in d_{max} and we produce a positive effect on the value of $\lambda_2(\mathbf{W}(\mathbf{G}'))$.

Therefore, by removing links (sparsifying the network), it is possible to create two opposite effects which, when well designed distributed methods are applied, the second positive effect can dominate over the first negative by making $\frac{\lambda_2(\mathbf{L})}{d_{\max}}$ as large as possible, which in turn amounts to make $\lambda_2(\mathbf{W})$ as small as possible [6]. Then, sparsifying a network can improve both the convergence rate and the power consumption of the associated consensus algorithm. Thus, we are looking for a distributed

[1] $\rho(\mathbf{A}) = \max_{\{1 \leq i \leq N\}} |\lambda_i(\mathbf{A})|$

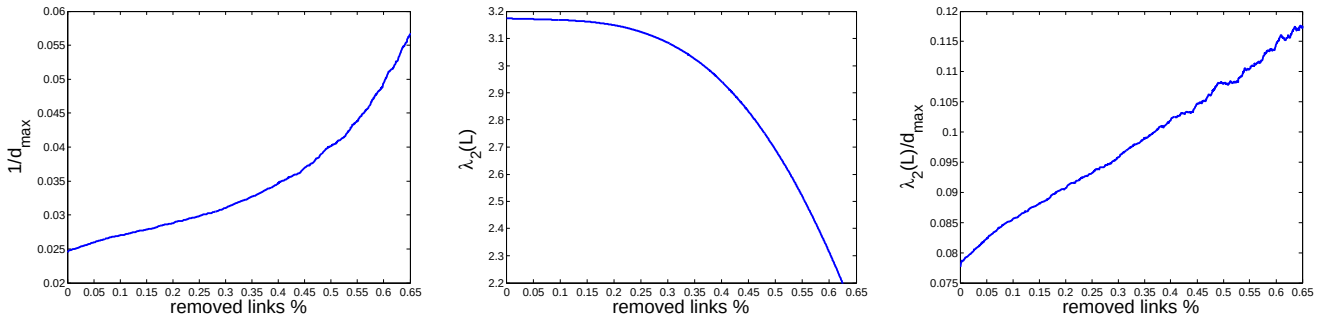

Figure 1: Average results over 100 different topologies that are randomly deployed and composed by 100 nodes. These graphs have been generated by using a greedy approach that removes, at each iteration, the link which minimizes $\lambda_2(\mathbf{W})$. The different parameters $\left\{\frac{1}{d_{max}}, \lambda_2(\mathbf{L}), \frac{\lambda_2(\mathbf{L})}{d_{max}}\right\}$, as a function of the percentage of removed links, are shown from left to right.

method which at the end of the process, reduces as much as possible the value of $\lambda_2(\mathbf{W})$.

As a initial idea for introducing our distributed algorithms, let us consider first the following centralized algorithm. Consider the idea of reducing $\lambda_2(\mathbf{W})$ iteratively by using a simple greedy approach that reduces the number on links in the network, that is, a simple process where the link to be removed, at each iteration step, is decided based on the criterion of minimizing this value. Figure 1 shows the results using this criterion. It is clearly shown that it is possible to reduce the value of $\lambda_2(\mathbf{W})$, thus speeding up the convergence of the consensus algorithm, by critically sparsifying (removing more than 60% of the links) a given dense network. However, this greedy approach is totally centralized and needs lot of computational resources and time to achieve a proper solution. Moreover, since it needs global information of the whole network, it is not scalable. Therefore, this paper focus on finding a distributed and scalable method for achieving similar results.

4. DISTRIBUTED LOCAL OPTIMIZATION ALGORITHM

The main idea behind this first distributed approach is that each node i, instead of having only information of its own neighborhood, we assume that it also knows the neighborhood of its one hop neighbors, thus it has local knowledge about the two hop subnetwork $\mathbf{SG}_i$ surrounding it. Having this extra information, the topology of this local subnetwork can be created at each node. Then, each node i can apply a convex optimization approach to the corresponding weight matrix $\mathbf{W}(\mathbf{SG}_i)$ in order to improve its spectral properties. Note that each link in a subnetwork is defined by a pair of nodes that also exists in the complete network. Then, disabling a link between a pair of nodes in the subnetwork also disables a link in the whole network.

The structure of the optimization subproblems is equivalent to the one used for optimizing the complete network, which is presented in [6]. However, we need to take into account two important differences: first, there exists an overlapping between the subnetworks and second, we need to fuse all the solutions in order to achieve the general solution. Moreover, since usually most of the links of the two-hop neighbors are not present in the subnetworks, their degree (within the subnetwork) is generally low and they are not good candidates to disable their links. Then, we introduce an additional constraint which consists on fixing their links to one, in order to reduce the number of variables, hence the complexity, in the optimization problem and the overlapping issue between the different solutions.

Then, each node i solves the following SDP problem:

$$
\begin{array}{ll}
\text{minimize}_{\{s,\mathbf{Z},\mathbf{Y}\}} & s \\
\text{s. t.} & \mathbf{W} = \mathbf{I} - \alpha(\mathbf{Z} - \mathbf{Y}) \\
& Y_{jk} = 0 \ \text{ if } \ (j,k) \notin \mathbf{E} \\
& Y_{jk} = 1 \ \text{ if } \ j \in \Omega_i \, , \ k \in \{\mathbf{SG}_i \backslash \{\Omega_i, i\}\} \\
& Y_{jk} \preceq 1, \ \ Y_{jk} \succeq 0 \\
& \mathbf{W} - \frac{\mathbf{1}\mathbf{1}^t}{N} \preceq s\mathbf{I} \\
& \mathbf{W}\mathbf{1} = \mathbf{1} \\
& \mathbf{1}^t\mathbf{W} = \mathbf{1}^t \\
& 1 \leq Z_{jj} \leq d_j \\
& \frac{1}{Z_{jj}} \geq \alpha
\end{array}
$$

Note that for the shake of simplicity, we use the notation of $\mathbf{W}$, $\mathbf{Z}$ for the variables instead of the corresponding $\mathbf{W}(\mathbf{SG}_i)$, $\mathbf{Z}(\mathbf{SG}_i)$. Moreover, we bound the maximum degree of the subnetwork with α, which takes values in the interval $\left[\frac{1}{d_{max}(\mathbf{SG}_i)}, \frac{1}{d_{max}(\mathbf{SG}_i)-1}, ...1\right]$.

We solve the previous subproblem for the different values of α. It gives an array of solutions where the best one is given by the minimum $\lambda_2(\mathbf{W})$, which implies minimum convergence time. This eigenvalue can be easily obtained by the nodes using some of the methods presented in [9][10]. Additionally, the solutions of the subproblems are projected [6] and combined. The only information used in the final solution is the corresponding to the links between the central node and its direct neighbors. Then, if the decision of removing a link is only taken by one of the two subnetworks where the link information is located, the conflict is solved based on the final degree of the involved nodes. In practice, the link is removed if their degree is greater or equal than 0.35 times the average degree of the network.

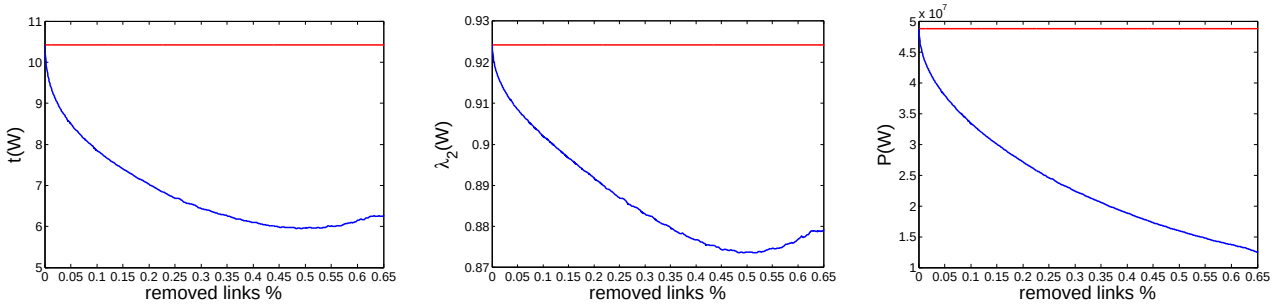

Figure 2: Average values of several parameters $\{t(\mathbf{W}), \lambda_2(\mathbf{W}), P(\mathbf{W})\}$ obtained by applying our power method and averaging over 100 different topologies composed by 100 nodes. The different parameters values are shown as a function of the percentage of removed links, from left to right: $t(\mathbf{W})$, $\lambda_2(\mathbf{W})$ and $P(\mathbf{W})$. The red flat line represents their value when no links are removed.

One inconvenient of this approach is that every node needs to solve a convex optimization problem whose size depends on the average degree. Moreover, as we have presented before, just having $\lambda_2(\mathbf{W})$ and applying a centralized greedy approach is possible to easily improve the convergence by sparsifying the original network. It is natural to ask whether the idea of using $\lambda_2(\mathbf{W})$, as a indicator for choosing the links to be removed, can be performed in a decentralized fashion.

5. DISTRIBUTED ALGORITHM BASED ON LOCAL POWER METHOD

Assuming, at each node, the two-hop information presented before, nodes can use one of the several existing methods [9][10] for approximating the spectra of its weight submatrix. One of them is the so-called power method which is briefly recalled here for convenience. Let $\mathbf{W}$ be the weight matrix defined in (2) which is a $N \times N$ diagonalizable symmetric matrix with a corresponding dominant largest eigenvalue $\lambda_1(\mathbf{W})$. Then, the power method consists on the following:

1. Let $\mathbf{y_0} = \mathbf{z_0}$ be any initial vector in $\mathbb{R}^n$ whose largest component is 1.
2. By repeating the following steps:
   - Compute $\mathbf{y_k} = \mathbf{W z_{k-1}}$
   - Let m_k be the component of $\mathbf{y_k}$ with the largest absolute value.
   - Set $\mathbf{z_k} = \frac{1}{m_k}\mathbf{y_k}$

The sequence of m_k converges to the dominant eigenvalue and $\mathbf{z_k}$ converges to a dominant eigenvector. There exists several methods for finding subsequent eigenvalues that are smaller in absolute value than a given one. We assume that the deflation method is used [9][10].

The main idea behind this second approach is that each node applies the power method to its corresponding weight two-hop neighborhood submatrix for quantifying how much is lost or gained by removing a link between the central node and one of its direct neighbors. Each node i can use Algorithm 1 in order to obtain a vector containing the quality of its links. Then each node applies locally the power method $|\Omega_i|$ times, each time corresponding to the removal of a one-hop link between the central node and a directly connected one. Since our goal is to minimize $\lambda_2(\mathbf{W})$, the links to be removed are the corresponding vector entries with minimum value.

Then, instead of trying all the possible combinations of removing 1,2...,etc. links we only remove one link each time and the same for all the possible one link removals in the first hop neighborhood of the central node. The computational cost required by this method only depends on the average degree of the whole network and not on the total number of nodes N, so it is scalable.

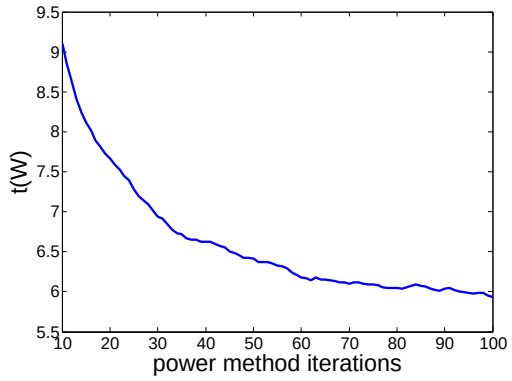

Figure 3: Consensus convergence time as a function of the precision (number of iterations) needed by the power method which is used by all nodes in parallel.

However, the precision of the eigenvalue approximation depends on the number of iterations used by the power method, so we need to analyze the cost needed for getting a good approximation in our specific problem. Figure 3 shows that few iterations of the power method are needed for getting good results, that is, nodes only need a few iterations of the power method (between 50 and 100 in practice) for getting good approximations of the corresponding eigenvalues and remove the correct links. Therefore, the total computational cost of the procedure is quite small.

Finally, a criterion for knowing how many links should be removed is necessary. Figure 2 shows that the optimal range is found when removing between 50% and 65% of the total number of links. Since, removing 65%

Algorithm 1 Neighborhood Quality vector

Require: $\mathbf{A}_{M\times M}, d_1, ..., d_M$
Ensure: $\mathbf{Q}$ contains $|\Omega_i|$ values of $\lambda_2(\mathbf{W})$
  for $j = 1 : ||\Omega_i||$ **do**
    $A(i,j) = 0,\ A(j,i) = 0$
    $D = \mathbf{diag}(d_1, d_i - 1, ..., d_j - 1, ..., d_M)$
    $\alpha = \frac{1}{\max(d_1, d_i-1, ..., d_j-1, ..., d_M)}$
    $\mathbf{W} = I - \alpha(\mathbf{D} - \mathbf{A})$
    $Q(j) = \lambda_2(\mathbf{W})$ {Obtained using the power method}
    $A(i,j) = 1,\ A(j,i) = 1$
  end for

of the links has similar convergence time than removing 50% of the links, we take this first value as a stoping criterion. Moreover, from this value, it can be easily found [6] the corresponding degree as a function of the average degree of the complete network and this value can be used by the nodes to compare it with its own degree and decide if it needs or not to lose more links. Our experimental results show that, this new method ensures that many more links are removed as compared to other previous methods presented in the literature. Thus, instead of only removing links between high degree nodes, as proposed in [6], we also allow to remove links between any pair of nodes based on $\lambda_2(\mathbf{W})$ approximations.

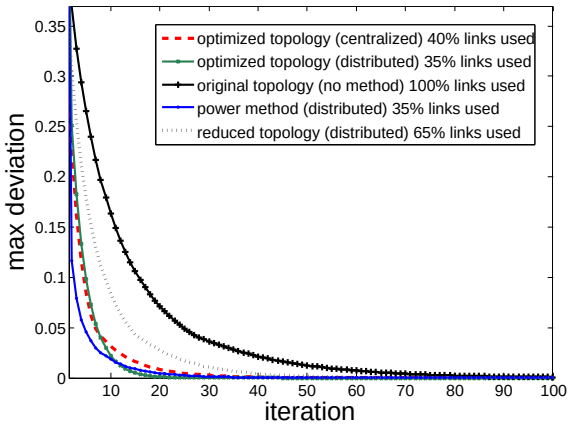

Figure 4: Average convergence results over 100 different topologies composed by 100 nodes. The comparison is done between the consensus executed in the topologies resulted from applying: a) our distributed optimization method, which only uses 35% of the links, b) our distributed power method, which also uses 35% of the links, c) and d) the two methods presented in [6] that use 65% and 40% of the links respectively and e) the original topology which uses 100% of the links.

6. NUMERICAL RESULTS

In this section, we focus on validating our algorithms by comparing them with previous work. Figure 2 shows the improvement of different parameters when applying the power method. It presents similar eigen-results than the centralized greedy approach, that is Figure 1. Although, the minimum convergence time is achieved when 50% of the links are removed, the time that is obtained by removing 65% of them is almost the same and it has much less power consumption. Then, this percentage is used on Figure 4, which shows a comparison of the convergence between our methods and the approaches presented in [6]. Our new methods present very good results in the initial interations. This is specially useful in detection applications where some quickly averaging is needed for taking a decision. Finally, Table 1 summarizes the numerical results of the corresponding parameters (time, power consumption and used links).

Table 1: Simulation Results

Method	$t(\mathbf{W})$	$P(\mathbf{W})$	used links %
original topology	10.2	4.7×10^7	100%
previous work (dist)	7.6	2.9×10^7	65%
previous work (cent)	5.5	1.7×10^7	40%
power method	6.25	1.3×10^7	35%
dist. optimization	6.5	1.5×10^7	35%

7. CONCLUSION

In this paper, we have tackled the problem of improving the average consensus problem in a distributed way. We have shown that it is possible to reduce the convergence time and the power consumption of this process by critically sparsifying a given network. The main contribution can be summarized as twice the improvement on link removal while having better convergence rate and power consumption in the associated average consensus method.

REFERENCES

[1] Lin Xiao; Boyd, S., "Fast linear iterations for distributed averaging," Decision and Control, 2003. Proceedings. 42nd IEEE Conference on , vol.5, no., pp. 4997-5002 Vol.5, 9-12 Dec. 2003.

[2] Xiao, L.; Boyd, S.; Lall, S., "A scheme for robust distributed sensor fusion based on average consensus," Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on , vol., no., pp. 63-70, 15 April 2005.

[3] Ming Cao; Chai Wah Wu, "Topology design for fast convergence of network consensus algorithms," Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on , vol., no., pp.1029-1032, 27-30 May 2007.

[4] Olfati-Saber, R., "Ultrafast consensus in small-world networks," American Control Conference, 2005. Proceedings of the 2005 , vol., no., pp. 2371-2378 vol. 4, 8-10 June 2005.

[5] Zhipu Jin; Murray, R.M., "Random consensus protocol in large-scale networks," Decision and Control, 2007 46th IEEE Conference on , vol., no., pp.4227-4232, 12-14 Dec. 2007

[6] Asensio-Marco, C.; Beferull-Lozano, B; "Accelerating consensus gossip algorithms: Sparsifying networks can be good for you,"IEEE ICC, 2010.

[7] C. Godsil and G. Royle. "Algebraic Graph Theory." Springer, 2001.

[8] M. E. J. Newman, "The structure and function of complex networks," SIAM Review, vol. 45, pp. 167-256, 2003.

[9] B. N. Parlett. "The Symmetric Eigenvalue Problem. Prentice-Hall," Englewood Clis, New Jersey, 1980.

[10] Y. Saad. "Numerical Methods for Large Eigenvalue Problems." Manchester University Press, Manchester, UK, 1992.