A Local Average Consensus Algorithm for Wireless Sensor Networks

Konstantin Avrachenkov Mahmoud El Chamie Giovanni Neglia
INRIA Sophia Antipolis - Méditerranée
2004 Route des Lucioles, B.P. 93
06902 Sophia Antipolis, France
{konstantin.avratchenkov,giovanni.neglia}@inria.fr, shamieh_m@yahoo.com

[1] ***Abstract*—In many application scenarios sensors need to calculate the average of some local values, e.g. of local measurements. A possible solution is to rely on consensus algorithms. In this case each sensor maintains a local estimate of the global average, and keeps improving it by performing a weighted sum of the estimates of all its neighbors. The number of iterations needed to reach an accurate estimate depends on the weights used at each sensor. Speeding up the convergence rate is important also to reduce the number of messages exchanged among neighbors and then the energetic cost of these algorithms. While it is possible in principle to calculate the optimal weights, the known algorithm requires a single sensor to discover the topology of the whole network and perform the calculations. This may be unfeasible for large and dynamic sensor networks, because of sensor computational constraints and of the communication overhead due to the need to acquire the new topology after each change. In this paper we propose a new average consensus algorithm, where each sensor selects its own weights on the basis of some local information about its neighborhood. Our algorithm is tailored for networks having cluster structure, like it is common for wireless sensor networks. In realistic sensor network topologies, the algorithm shows faster convergence than other existing consensus protocols.**

I. Introduction

Emerging technologies as robotics, multi-vehicle cooperation control, and environmental monitoring have a driving need for wireless sensor networks. In these scenarios, sensors often need to reach consensus, for example for sensor fusion, coordination or optimization of algorithm parameters. We are interested in this paper in average consensus algorithms [1], where sensors deployed in a network, each having an initial value, e.g. a measurement, aim to calculate the average of all these values through a distributed linear iteration method. Consider for instance sensors deployed to measure the temperature T of a given source. Due to additive zero mean Gaussian noise, each sensor has a different measurement of the source temperature. It is well known that a good filter of the Gaussian noise is the mean filter. Therefore, if we average the values of the initial measurements we can have a good estimation of the source temperature. The advantage of consensus algorithms is that they can calculate iteratively this average value in a completely distributed way through local information exchange among neighbors and simple calculation of weighted sums at each sensor. The speed of convergence of consensus algorithms depends on the weights used by each sensor. This is a critical performance metric also for energetic reasons. In fact reducing the convergence time leads to a smaller number of transmissions among the sensors and then to lower energetic costs for each sensor. It is known that the set of weights that minimize the convergence time can be determined through a Semi Definite Program ([2], [3]) that requires the knowledge of the whole topology. This approach may be unfeasible for large and dynamic sensor networks, because of sensor computational constraints and of the communication overhead due to the need to acquire the topology after each change. In this scenario, it seems more promising to use *local average consensus algorithms*, i.e. algorithms where each node autonomously calculates its weights on the basis of some local knowledge.

In this paper we propose a new algorithm, called neighborhood algorithm, that requires less iterations than other existing consensus algorithms on different types of graphs having cluster structure. On these graphs the speed of convergence is greatly affected by the weights given to the links connecting different clusters. The neighborhood algorithm has been designed to identify such links and give them higher weights in order to speed-up information propagation among different parts of the networks.

The outline of the paper is as follows. Section II provides a background on average consensus algorithms, describing also convergence conditions and the specific algorithms considered in this paper for comparison purpose. We describe our consensus algorithm in section III and we argue that it converges faster than other known algorithms on networks having cluster structure. In section IV we shows simulation results that support this claim on different graph topologies such as 2-cliques, Watts-Strogatz and random geometric graphs. Finally section V summarizes the paper and discusses future work.

II. Average Consensus Algorithms

The study of average consensus algorithms has gained a lot of interest in the last decade [1], [4], [5], [6]. We may divide these algorithms into two main classes: synchronous and asynchronous algorithms. In asynchronous algorithms, each node in the network has an independent clock. When the node's

[1]The authors are given in alphabetical order.

clock ticks, the algorithm is applied by the node independently from any other node. For a complete coverage of asynchronous algorithms we recommend to read [7], [8], [9], [10], [11], [12]. On the other hand, in synchronous algorithms, all nodes in the network use a global clock and perform every iteration of the average consensus algorithm at the same time. In this paper we only consider synchronous algorithms on static topologies, but the rationale of our algorithm can be extended also to the asynchronous case and, since the algorithm uses only local information to select the weights, it can also operate in a dynamic scenario.

In order to describe the operation of synchronous consensus algorithms, we need to introduce some notation. We consider an undirected graph $G = (V, E)$ where the vertices in $V = \{1 \dots n\}$ correspond to the nodes in the network, and an edge $(i, j) \in E$ corresponds to a communication possibility between nodes i and j. Let N_i be the set of neighbors of node i and $N_{[i]}$ be the closed neighborhood, i.e. $N_{[i]} = \{i\} \cup N_i$. Each node i initially has a local value $m_i \in \mathbb{R}$. The purpose of an average consensus algorithm is to have nodes calculating the average $\bar{m} = \frac{1}{n}(\sum_{i=1}^{n} m_i)$ in an iterative and distributed way. To this purpose, each node maintains a local estimate of the global average $\bar{m}$. Let us denote by $x_i(k)$ the estimate of node i at time slot k such that $x_i(0) = m_i$. At time slot $k+1$, each node transmits its current estimate to its neighbors and updates its estimate as follows: $x_i(k+1) = \sum_{j \in N_{[i]}} w_{ij} x_j(k)$, or in matrix form:

$$\mathbf{x}(k+1) = W\mathbf{x}(k) \tag{1}$$

where W is the matrix whose elements are the weights w_{ij} and $\mathbf{x}(k)$ is the state vector of the system at time slot k, i.e. the vector whose elements are the node estimates $x_i(k)$.

An average consensus algorithm has to guarantee that the iterative procedure described by eq. (1) leads all the estimates to converge to $\bar{m}$, i.e.:

$$\lim_{k \to \infty} \mathbf{x}(k) = \bar{\mathbf{m}} \triangleq \bar{m}\mathbf{1},$$

where $\mathbf{1}$ is a column vector of all ones. In [2] Xiao and Boyd showed that the necessary and sufficient conditions for the convergence in the case of static networks are:

$$W\mathbf{1} = \mathbf{1}, \tag{2}$$

$$\mathbf{1}^T W = \mathbf{1}^T, \tag{3}$$

$$\rho\left(W - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) < 1, \tag{4}$$

where $\rho(A)$ is the spectral radius of matrix A.

Equation (2) means that the consensus is stable, i.e. if $\mathbf{x}(k_f) = \bar{\mathbf{m}}$ then $\mathbf{x}(k) = \bar{\mathbf{m}}\ \forall k > k_f$. Equation (3) says that the average of all the estimates is conserved at each iteration which guarantees the final consensus value to be the average of the initial measurements. Finally, equation (4) is the contraction condition, i.e. it corresponds to the fact that the distance of the estimates from $\bar{m}$ is decreasing.

Even if the matrix W has to satisfy these conditions in order to guarantee the convergence of the estimates to $\bar{m}$, the choice of the weights is to a given extent arbitrary and different algorithms differ in the way weights are selected. In particular the matrix can be selected to speed up the convergence. The convergence speed of these algorithms is related to the mixing time of Markovian chains, it is possible to show that the "error" $||\mathbf{x}(k) - \bar{\mathbf{m}}||_2$ can be upper bounded by $c\lambda_2^k$, where λ_2 is the second largest eigenvalue of the weight matrix W and c is a constant [3], [13]. In [2], [3], the authors address the problem of selecting the optimal weights and formulate a semidefinite program to determine the matrix W with the smallest value of λ_2. If the network topology is not known a priory (or changes in time), the implementation of this optimization procedure requires a single sensor to discover the topology of the whole network and perform the calculations. This may be unfeasible for large and dynamic sensor networks, because of sensor computational constraints and of the communication overhead due to the need to acquire the topology and recalculate all the weigths after each change. In these cases, it seems better to let each sensor to calculate independently its weights on the basis of some local knowledge. In this way the computation burden for each node is smaller and topology changes at one location (e.g. the failure of a link or of a node) only affect the weights of the nearby sensors.

One of the most known local average consensus algorithm is the **Local Degree algorithm** (also known as Metropolis-weight algorithm), where each sensor selects its weights on the basis of its own degree and the degree of its neighbors. In particular it holds

$$w_{ij} = \begin{cases} \frac{1}{\max\{d_i, d_j\}} & \text{if } (i,j) \in E \text{ and } i \neq j, \\ 0 & \text{if } (i,j) \notin E \text{ and } i \neq j, \end{cases} \tag{5}$$

where d_i (d_j) is the degree of vertex i (j). Then nodes select the weight w_{ii} such that $w_{ii} = 1 - \sum_{j \in N_i} w_{ij}$ in order to satisfy condition (2). The three convergence conditions are satisfied.

Two other common algorithms rely instead on some global knowledge of the network. Under the **Max Degree algorithm** each node is assumed to know the maximum node degree in the network ($d_{\max} = \max_i\{d_i\}$) and the same weight is associated to each link as follows:

$$w_{ij} = \begin{cases} \frac{1}{d_{\max}} & \text{if } (i,j) \in E \text{ and } i \neq j, \\ 0 & \text{if } (i,j) \notin E \text{ and } i \neq j. \end{cases} \tag{6}$$

The weights w_{ii} are then determined as above in order to guarantee the stochasticity of the matrix W. Also in this case convergence conditions are satisfied, but convergence can be very slow. It is possible to increase the convergence speed by increasing the constant weight value, but the risk is that condition (4) is no more satisfied. The optimal constant weight value is used by the **Best Constant algorithm**, that is the fastest one among those that use uniform weights. [2] proves that each weight w_{ij} with $i \neq j$ has to be selected equal to $\frac{2}{\nu_1(L)+\nu_{n-1}(L)}$, where L is the Laplacian of the graph and ν_i is the i^{th} largest eigenvalue.

III. The Neighborhood Algorithm

In local algorithms, each node selects its corresponding weights on the basis of local information without the need to know the whole network topology. Our algorithm, called Neighborhood algorithm, falls within this category and is tailored for networks with cluster structure, i.e. networks where nodes tend to create tightly knit groups characterised by a relatively high density of links. Sensor networks usually exhibit a cluster structure. In fact, since available links are determined by sensors being in the transmission range of each other, there is a big chance that a node and its neighbor have many common neighbors and form a cluster.

In general, graphs with cluster structure require a large number of iterations for consensus algorithms to converge because information tends to remain confined within each cluster and it slowly propagates from a cluster to another. For example, in the network in Fig. 1 the local estimates in each cluster converge fast to the average of the initial values in the cluster, but they slowly reach the global average because communication between the two clusters is possible only through the link (u, v). The example shows that in networks with cluster structure, links have significantly different roles in the spreading of the information. Intuitively, a way to speed up the convergence is to give higher weights to links that are more important because they connect different parts of the networks (like the link (u, v)). The rationale of our algorithm is indeed to let each node identify such links relying only on some local knowledge. In particular in the Neighborhood algorithm, the importance of the links is estimated locally by considering the neighborhood characteristics of nodes in the network. Each node i sets the weight of a link (i, j) depending on the similarity between its neighborhood set and the neighborhood of node j. In order to quantify such similarity we resort to the Jaccard index defined in the set theory [14]. For any two sets A, and B, the Jaccard index is:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{7}$$

The smaller $J(N_{[i]}, N_{[j]})$ (we will refer to it as J_{ij}), the more different the two neighborhood sets. Then the corresponding link (i, j) is given a higher weight. The larger J_{ij}, the more similar the neighborhoods and the smaller the link weight. For example, u and v in Fig. 1 must give higher weight to the link (u, v) than to any other link as the two nodes have very different neighborhood sets.

Algorithm 1 describes the detailed operation of the Neighborhood algorithm. Nodes must first exchange their neighborhood sets and then select the weights according to the similarities between the neighborhoods. The operations in steps 3 and 5 of the algorithm guarantee that the weights satisfy the convergence conditions given in the section II. In fact, the normalization in step 3 makes possible to determine a weight matrix W stochastic, while taking the minimum between the two values y_{ij} and y_{ji} at step 5 generates a symmetric matrix. Finally, the way to select weights w_{ii} guarantees that the matrix is stochastic and then, being that it is symmetric, doubly stochastic.

Algorithm 1 Neighborhood Algorithm

1: Each node i broadcasts the set $N_{[i]}$ to all its neighbors.
2: After receiving all the sets from its neighbors, node i assigns to each link (i, j), $j \in N_i$ the value:

$$y_{ij} = 1 - \frac{|N_{[i]} \cap N_{[j]}|}{1 + 2 \times \min\{|N_{[i]}|, |N_{[j]}|\} - |N_{[i]} \cap N_{[j]}|}. \tag{8}$$

3: $a = \sum_{j \in N_i} y_{ij}$
if $a > 1$ then $y_{ij} := y_{ij}/a$.
4: Each node i sends to every neighbor j the value y_{ij}.
5: Each node i sets $w_{ij} = \min\{y_{ij}, y_{ji}\}$ $\forall j \in N_i$ and $w_{ii} = 1 - \sum_{j \in N_i} w_{ij}$.

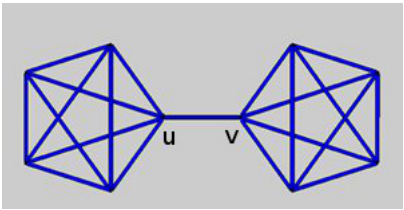

Fig. 1. Example of a 2-clique graph with $n = 10$ nodes.

IV. Performance Evaluation

We have considered three different types of networks to compare the speed of convergence of the different average consensus algorithms:

- 2-Clique Graph: On these graphs, 2 cliques each having $n/2$ nodes are connected by only one link, see Fig. 1. The characteristic of these graphs is the existence of a bottleneck link. For the nodes to reach consensus, information must flow through the link joining the two clusters. Thus, the weight of this link is expected to play an important role in the convergence of average consensus algorithms.
- Watts-Strogatz (WS) Graphs [15]: Small world graphs defined by the number of nodes n, their average degree, and a rewiring probability p, see Fig. 2. When $p = 0$, each node on the graph has high clustering coefficient, however, as p increases, the graph becomes more random, and both the diameter and the clustering coefficient become smaller. We will study the effect of the rewiring probability on the speed of convergence of average consensus algorithms.
- Random Geometric Graphs (RGG) [16]: Graphs where n nodes are placed uniformly at random on a convex unit area (we will consider a unit square area), and any two nodes are connected by an edge if the distance between them is less than the radius $r_n = \sqrt{c \times \frac{\log(n)}{n}}$, where c is a constant, see Fig. 3. RGG are well suited to model wireless sensor networks

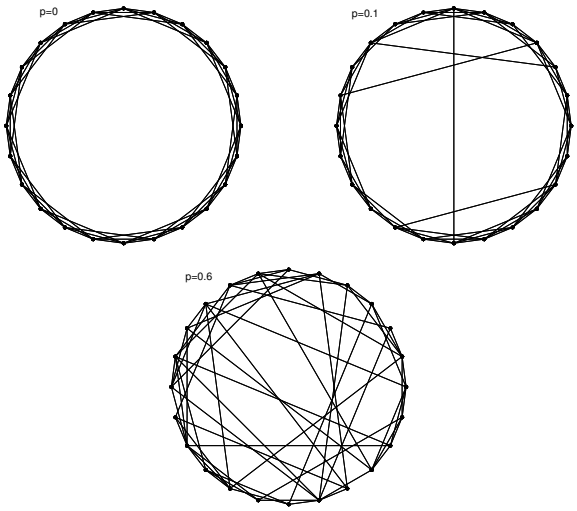

Fig. 2. Example WS graph with $n = 24$ nodes, average degree 6, and p equals 0, 0.1, and 0.6.

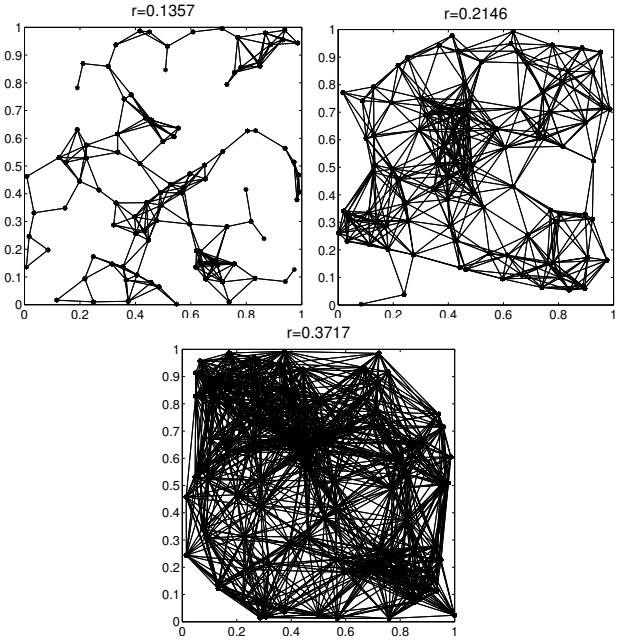

Fig. 3. RGG with $n = 100$ nodes and different values of the connectivity radius.

where the nodes have been deployed randomly on a field and the transmission range of each sensor is r_n. When the transmission range r_n is small, the network presents clusters of sensors, connected by links that play an important role for the convergence. As the range is increased, the network becomes more connected and converges to a complete graph. We have studied the convergence speed of the average consensus algorithms for different transmission range values.

For all our simulations, we considered the graph to have $n = 100$ nodes.

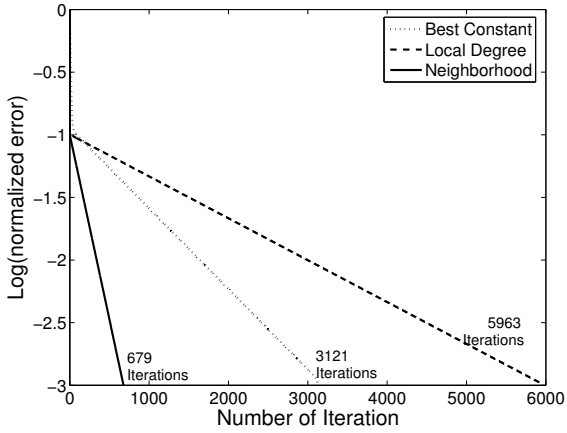

Fig. 4. Convergence speed on 2-Clique graphs.

The distance of the system state vector $\mathbf{x}(k)$ from the average $\bar{\mathbf{m}}$ can be expressed by the normalized error:

$$e(k) = \frac{||\mathbf{x}(k) - \bar{\mathbf{m}}||_2}{||\mathbf{x}(0) - \bar{\mathbf{m}}||_2}. \quad (9)$$

Figure 4 shows the performance of the algorithms on 2-clique graphs in terms of the normalized error. The Max Degree algorithm is not shown as its performance is very similar to the Local Degree one. As initial condition we considered one node with initial value equals to 1 and all other nodes with initial value equal to 0. The estimate of each sensor will then converge to the value $\bar{m} = 1/n = 0.01$ with a different speed depending not only on the algorithm but also on the specific node that has value 1. We have then averaged the error at each iteration across all possible permutations of the initial values. Our algorithm converges much faster than the other two. In particular, the error becomes smaller than 10^{-3} after 679 iterations while more than 3000 iterations were required for the other algorithms. Thus the neighborhood algorithm on 2-clique graph is able to identify the importance of the bottleneck link (u, v), and as u and v have totally different neighborhoods, more weight is given to the link (u, v) than to any other link in the graph.

The two following metrics are used to compare the speed of convergence of the average consensus algorithms on RGG and WS graphs.
The convergence factor is defined as:

$$\nu(W) = \sup_{\mathbf{x}(0)\neq\bar{\mathbf{m}}} \lim_{k\to\infty} e(k)^{1/k} = \rho\left(W - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right), \quad (10)$$

where $\rho(A)$ is the spectral radius of matrix A. The convergence time is defined as:

$$T_{conv} = \frac{1}{\log(1/\nu)}. \quad (11)$$

It is roughly the time for the normalized error to become smaller than $1/e$.

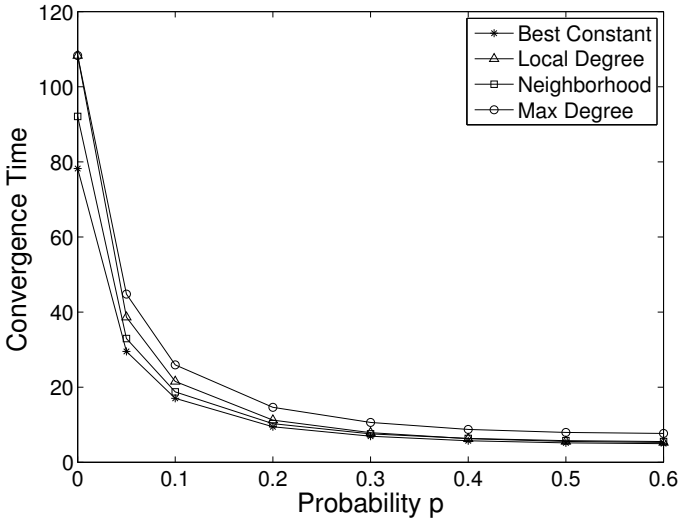

Fig. 5. Convergence time on Watts Strogatz graphs.

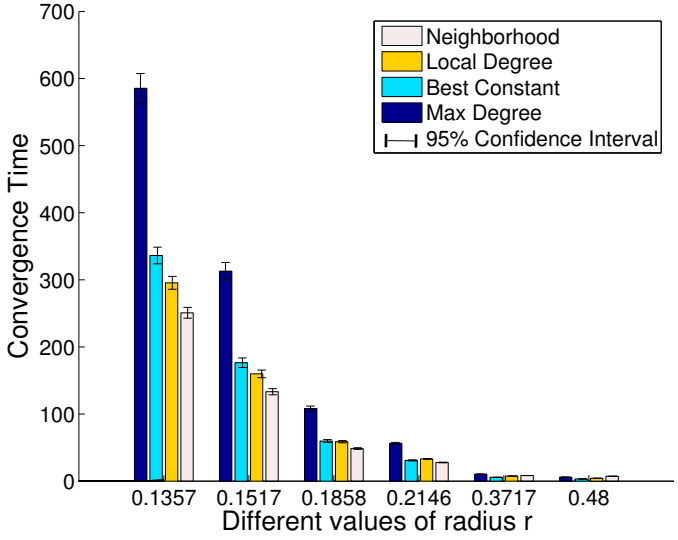

Fig. 6. Convergence time on RGG graphs.

Figure 5 shows the average convergence time over 1000 WS graphs with average degree 6, for different values of the rewiring probability p between 0 and 0.6. For WS graphs, $p = 0$ gives high clustered graphs; however, as the value of p increases, the graph becomes more random, and the diameter of the graph decreases significantly, thus giving a random graph with reasonable convergence time for average consensus. For small values of probability p, the Neighborhood algorithm performs better than the Local Degree and the Max Degree algorithms. In particular the Neighborhood algorithm saves up to 15% of the iterations as compared to the other two for $p \leq 0.1$. The Best Constant algorithm performs even better, but it requires, as we said, the global knowledge of the network. For larger values of p, almost all the algorithms achieve similar performance and in particular they have much smaller convergence times.

Similarly, Fig. 6 shows the average convergence time of the different algorithms over 1000 RGG graphs for different values of the transmission range r. For small values of r, the Neighborhood algorithm gives the best performance among the considered algorithms, in this case even better than the Best Constant algorithm. As RGG graphs with small r present cluster structure, some links have a more important role for convergence and setting a constant weight on all the links gives a longer convergence time. It appears that the level of similarity between the neighborhood sets (used by our algorithm) is a good estimator of the correct weights of the links and this is shown by the faster convergence of the Neighborhood algorithms for a radius $r \leq 0.2146$. For example, on RGG with radius $r = 0.1357$, the convergence time for Neighborhood algorithm is 245, it reaches 590 for the Max Degree algorithm, while the second shortest time is obtained by the Local Degree algorithm that needs about 295 iterations. Thus the Neighborhood algorithm saves approximately 17% of the iterations and this results in 17% less messages in the network. This improvement can be very important in sensor networks because power consumption is a critical issue. As we increase the transmission range, the graph becomes more connected and convergence is guaranteed by any algorithm within a reasonable number of iteration. Large transmission ranges are in any case probably not realistic for sensor networks because they would cause a lot of interference. For example with $r \geq 0.3$ a sensor interferes on the average with 25% of the other sensors.

V. Conclusion

In this paper we proposed a new average consensus algorithm for wireless sensor networks, called Neighborhood algorithm. As other local consensus algorithms, our algorithm can operate in large dynamic networks because every sensor calculates its weights on the basis of some local information. The algorithm has been tailored for networks with cluster structure, where there are groups of nodes well connected among themselves but poorly connected with other groups. In such scenario, consensus algorithms can achieve poor performance and a large convergence time. Our algorithm aims to identify the most important links for information propagation to give them a higher weight and then speed up the convergence. Simulation results in section IV show that indeed the Neighborhood algorithm outperforms existing local algorithms and even the Best Constant algorithm in Random Geometric Graphs that are a quite realistic model for many sensor networks.

Future research is going to follow two directions. First, we want to provide some analytical upper bound to the convergence time of the Neighborhood algorithm. Our main idea is that it is possible to provide a bound on the convergence time of a consensus algorithm in terms of the conductance of the weighted graph corresponding to the matrix W. Then, it is possible to interpret the operation of the Neighborhood

algorithm (or of similar variants) as an attempt to maximize the local conductance in the neighborhood of each node. Second, in this paper we have only considered synchronous algorithms on static topologies, but the rationale of our algorithm can be extended also to the dynamic network scenario, being that the algorithm uses only local information to select the weights. Then a next step is to extend the Neighborhood algorithm to that case and to study its behaviour.

REFERENCES

[1] R. Olfati-saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proceedings of the IEEE*, January 2007.

[2] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2004.

[3] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM REVIEW*, vol. 46, pp. 667–689, April 2004.

[4] R. Olfati, S. Richard, and M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, pp. 1520–1533, September 2004.

[5] A. Nedi, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Trans. Autom. Control*, vol. 54, November 2009.

[6] S. Kar and J. M. F. Moura, "Topology for global average consensus," *in 40th Asilomar Conference on Signals, Systems, and Computers*, October 2006.

[7] P. Denantes, F. Bénézit, P. Thiran, and M. Vetterli, "Which distributed averaging algorithm should i choose for my sensor network?" *27th IEEE Conf. Computer Communications and Networks*, April 2008.

[8] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2508–2530, June 2006.

[9] F. Bénézit, A. G. Dimakis, P. Thiran, and M. Vetterli, "Order-optimal consensus through randomized path averaging," *IEEE Trans. Inf. Theory*, vol. abs/0802.2587, October 2008.

[10] B. Nazer, A. G. Dimakis, and M. Gastpar, "Neighborhood gossip: Concurrent averaging through local interference," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 3657–3660, April 2009.

[11] A. G. Dimakis, A. D. Sarwate, and M. Wainwright, "Geographic gossip : Efficient aggregation for sensor networks," *5th International Symposium on Information Processing in Sensor Networks*, April 2006.

[12] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Trans. Signal Process.*, vol. 57, pp. 2748–2761, July 2009.

[13] J. S. Rosenthal, "Convergence rates for markov chains," *SIAM Rev.*, vol. 37, pp. 387–405, September 1995.

[14] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[15] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, June 1998.

[16] M. Penrose, *Random Geometric Graphs*. Oxford Studies in Probability, 2003.