Joint 48th IEEE Conference on Decision and Control and
28th Chinese Control Conference
Shanghai, P.R. China, December 16-18, 2009

Multiscale networks for distributed consensus algorithms

Christina Selle* and Matthew West†

Abstract— **We introduce a multiscale network construction that accelerates the convergence of distributed consensus algorithms on the network. Local update rules are given to account for node and edge failure, and the trade-off between performance and robustness of the multilevel network is investigated. A numerical example is provided to demonstrate the improved convergence rate obtained using the new algorithm.**

I. INTRODUCTION

Distributed consensus algorithms [4][10][17] allow a network of computational nodes to iteratively exchange information between neighbors in order to compute the global average of a quantity. They can be used as the basis for many applications, such as distributed optimization methods [6] or control schemes [3]. While typically less efficient than a centralized algorithm, consensus methods have the advantages of distributing the work across all nodes in the network and of being robust to node and connection failure.

The general framework for consensus methods considers each node synchronously updating its own value to a weighted average of the current values of its neighbors (as distinct from asynchronous gossip algorithms [2], for example). One of the most natural questions, therefore, is what graph structure and what weights should be chosen to give the fastest convergence of the algorithm to the consensus value.

The choice of optimal weights has been investigated in depth by [1][13], who used convex optimization and semidefinite programming to find the weights that minimize the magnitude of the second largest eigenvalue of the Markov chain defined by the consensus update. While such an approach gives the optimal choice of weights, it requires a centralized scheme for solving the optimization problem for the weights.

An alternative to solving for the optimal weights is to choose a graph structure that gives fast consensus with some weight choice. This can be done by optimization [5], or by using a heuristic such as taking advantage of the fact that small-world networks [16] have fast consensus [14] and thus trying to add edges or nodes to enhance this property. Other possibilities include making the node updates random [7] or otherwise time-varying [8].

In this paper we present an alternate scheme for producing a network to achieve fast consensus, based on the idea of *multiscale networks*. We observe that a regular consensus method is similar to using Jacobi's method to solve the equation $Lx = 0$, where L is the graph Laplacian or a similar matrix. Unfortunately, the convergence rate of Jacobi's method is poor and scales badly as system size grows [12]. This is due to the fact that errors that vary slowly across the network are only slowly driven to zero by the Jacobi iteration, which uses only nearest-neighbor updates. One standard way of overcoming these deficiencies is to use multilevel algorithms, such as the multigrid method [15], where coarsened versions of the base-level graph are used to enhance the decay of slowly varying components.

We build on this insight and give an algorithm for constructing multilevel networks for consensus. The basic multilevel network construction is presented in Section II, with a heuristic for adjusting the weights to enhance convergence in Section III. An algorithm for adjusting the edge weights in the presence of node and edge failures is given in Section IV and the trade-off between performance and robustness is investigated numerically in Section V. Finally, Section VI presents a numerical example for a randomly generated graph embedded in 2D.

II. CONSTRUCTION OF A MULTILEVEL NETWORK

By a multilevel network we mean one where nodes are arranged in levels or classes. All nodes are not equal in their connection structures, but are grouped. In a spatially embedded network, lower levels contain more nodes and have physically short-range connections, while higher levels contain fewer nodes that have longer-range connections. This thus mimics the multiscale structure generated by multilevel algorithms such as multigrid [15]. We refer to nodes in all upper levels as "supernodes", to distinguish them from the nodes in the base level.

We start with a network with a set of nodes $\mathcal{N}$ and a set of edges $\mathcal{E}$ connecting these nodes. Each node is given an initial value, and the purpose of the consensus algorithm is to find the mean of the initial states of all nodes. The initial values of all nodes are stored in the vector $x(0)$. Starting with the initial values, at any time step t each node i takes a weighted average of the state values of its neighboring nodes to compute its own new state value $x_i(t+1)$. This process can be represented as a multiplication with a state transition matrix P:

$$x(t+1) = P^T x(t) \tag{1}$$

For a single-level network, Metropolis weights can be used to propagate the state as described in [17]. With Metropolis

*C. Selle is at the Department of Aeronautics and Astronautics, Stanford University, cmester@stanford.edu

†M. West is at the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, mwest@illinois.edu

weights, the state transformation matrix is

$$P_{ij} = \begin{cases} \frac{1}{1+\max d_i, d_j} & \text{if } \{i,j\} \in \mathcal{E} \\ 1 - \sum_{\{i,k\} \in \mathcal{E}} P_{ik} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

This is equivalent to probability distributions in Markov Chains and we assume irreducibility and aperiodicity so the state converges to a final state π, where

$$P^T \pi = \pi \quad (3)$$

State transition matrices that result from applying Metropolis weights are symmetric, and all row- and column-sums are equal to one. The invariant distribution is uniform, and represents the average of the initial states of the nodes. Metropolis weights can be computed quickly by the distributed network, and can be efficient for single-level networks. However, they result in inefficiencies when applied to multilevel networks. In particular, Metropolis weights for connections between supernodes in upper levels of the network are smaller than they would need to be to maximize the convergence rate, since Metropolis weights take into account only the degree of a node, but not other aspects of the geometry of the network, such as the length of edges in a spatial embedding.

One method for constructing multilevel networks and finding their state transition matrices and invariant distributions is to first generate the base level, and then add the upper levels. Each superior level is generated by making an identical copy of the next lower level, and merging several nodes into supernodes. The nodes in each level are connected to their equivalent nodes in the levels directly above and below.

This method can be used for constructing multiscale networks based on an arbitrary layout of the base-level network. It does however constrain the construction of the upper levels and the connections between levels, in that connections between supernodes must mirror the connections in the lowest level. It can therefore be applied in situations where the geometry of the upper levels of the network can be chosen to fit these constraints, or when the layers of supernodes are created by selecting some of the regular nodes in the base level to double as supernodes, and the base-level connections between nodes are also used to implement supernode edges.

The first step in creating the multilevel network is duplicating the base level to create upper levels, and connecting each node to its corresponding node in the levels directly above and below. The connections between different levels initially all have equal weights going up and down. For such a network, the invariant distribution of each level is equal to the invariant distribution π of the original base level, so that the overall invariant distribution is

$$\hat{\pi} = \frac{1}{n} \left[\pi^T, \pi^T, \ldots, \pi^T\right]^T \quad (4)$$

Next, weights are added for connections between different levels, so that the values a node receives from superior levels can be given more weight than those from inferior levels. Using coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$ to denote weights for connections between nodes in each level, and $\beta_1, \beta_2, \ldots, \beta_{n-1}$ for weights of connections between levels, the new state transition matrix is

$$\hat{P} = \begin{bmatrix} \alpha_1 P & (1-\alpha_1)I & 0 & \cdots & 0 \\ \beta_1 I & \alpha_2 P & (1-\alpha_2-\beta_1)I & \cdots & 0 \\ 0 & \beta_2 I & \alpha_3 P & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_n P \end{bmatrix} \quad (5)$$

The merging of nodes to form q supernodes from p nodes in a level is described by the transformation matrix $B_i \in \Re^{p \times q}$, where $B_{ij} = 1$ if and only if the original node i is merged in supernode j. B_i thus describes the mapping from the base level of nodes to level i. Note that $B_1 = I$. The transformation from the ladder network to the final network is described by

$$\hat{B} = \text{diag}(B_1, B_2, \ldots, B_n) \quad (6)$$

$$\bar{\hat{P}} = \left(\hat{B}^T \hat{B}\right)^{-1} \hat{B}^T \hat{P} \hat{B} = \hat{B}^\dagger \hat{P} \hat{B} \quad (7)$$

Theorem 2.1: A multilevel network constructed as described above will have the state transition matrix (8) in figure 1 and invariant distribution $\bar{\hat{\pi}}$ given by

$$\bar{\hat{\pi}} = \left[(\gamma_1 \pi)^T, (\gamma_2 B_2^T \pi)^T, (\gamma_3 B_3^T \pi)^T, \ldots, (\gamma_n B_n^T \pi)^T\right]^T, \quad (11)$$

where the coefficients $\gamma_1, \gamma_2, \ldots, \gamma_n$ are found by solving the linear system (9) shown in figure 2.

Proof: Using $\hat{P}$ from eq. 5 in eq. 7 yields the state transition matrix shown in figure 1. Given $\hat{P}$, we can show that $\bar{\hat{\pi}}$ in equation 11 is indeed the invariant distribution:

$$\bar{\hat{P}}^T \bar{\hat{\pi}} = \begin{bmatrix} \alpha_1\gamma_1\pi + \beta_1\pi \\ ((1-\alpha_1)\gamma_1 + \alpha_2\gamma_2 + \beta_2\gamma_3) B_2^T \pi \\ \vdots \\ ((1-\alpha_{i-1}-\beta_{i-2})\gamma_{i-1} + \alpha_i\gamma_i + \beta_i\gamma_{i+1}) B_i^T \pi \\ \vdots \\ ((1-\alpha_{n-1}-\beta_{n-2})\gamma_{n-1} + \alpha_n\gamma_n) B_n^T \pi \end{bmatrix} = \bar{\hat{\pi}} \quad (12)$$

Since the α and β coefficients are known, this can be written as a system of linear equations. Omitting the last row, which is redundant since each column of the original system sum to zero, and adding the condition that the sum of the γ's has to be one, we get

$$\begin{bmatrix} (\alpha_1 - 1) & \beta_1 & \cdots & 0 \\ (1-\alpha_1) & (\alpha_2 - 1) & \cdots & 0 \\ 0 & (1-\alpha_2-\beta_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_{n-1} \\ 1 & 1 & \vdots & 1 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_{n-1} \\ \gamma_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \quad (13)$$

The linear system in figure 2 is constructed by taking the sum of each row except the last with all rows above it. As long as $\alpha_i + \beta_{i-1} < 1$ for all i, there is a unique solution. Given this solution, each node can determine the consensus value from the invariant distribution. ■

The resulting invariant distribution is not uniform, and in order to determine the consensus value, the state of each node has to be multiplied with a factor that can be obtained by solving the linear equation above for the invariant distribution.

$$\bar{\tilde{P}} = \begin{bmatrix} \alpha_1 P & (1-\alpha_1)B_2 & 0 & \cdots & 0 & 0 \\ \beta_1 B_2^\dagger & \alpha_2 B_2^\dagger P B_2 & (1-\alpha_2-\beta_1)B_2^\dagger B_3 & \cdots & 0 & 0 \\ 0 & \beta_2 B_3^\dagger B_2 & \alpha_3 B_3^\dagger P B_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & (1-\alpha_n)B_n^\dagger B_{n-1} & \alpha_n B_n^\dagger P B_n \end{bmatrix} \tag{8}$$

Fig. 1. Transition matrix for Theorem 2.1. $B^\dagger$ denotes the pseudoinverse of B, i.e. $B^\dagger = \left(B^T B\right)^{-1} B^T$

.

$$\begin{bmatrix} (\alpha_1-1) & \beta_1 & 0 & \cdots & 0 & 0 \\ 0 & (\alpha_2+\beta_1-1) & \beta_2 & \cdots & 0 & 0 \\ 0 & 0 & (\alpha_3+\beta_2-1) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & (\alpha_{n-1}+\beta_{n-2}-1) & \beta_{n-1} \\ 1 & 1 & 1 & \vdots & 1 & 1 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_1+\gamma_2 \\ \gamma_1+\gamma_2+\gamma_3 \\ \vdots \\ \sum_{i=1}^{n-1}\gamma_i \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{9}$$

Fig. 2. Linear system for Theorem 2.1.

$$\bar{\tilde{P}}_a = \begin{bmatrix} \alpha_1 P & (1-\alpha_1)B_2 & 0 & \cdots & 0 \\ \beta_1 B_2^\dagger & \alpha_2\left((1+\delta_2)B_2^\dagger P B_2 - \delta_2 I\right) & (1-\alpha_2-\beta_1)B_2^\dagger B_3 & \cdots & 0 \\ 0 & \beta_2 B_3^\dagger B_2 & \alpha_3\left((1+\delta_3)B_3^\dagger P B_3 - \delta_3 I\right) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_n\left((1+\delta_n)B_n^\dagger P B_n - \delta_n I\right) \end{bmatrix} \tag{10}$$

Fig. 3. State transition matrix with adjusted supernode self-weights.

III. Adjusting weights for improved performance

The method for constructing the propagation matrix has one deficiency: since supernodes are constructed by merging several nodes in one level into one supernode, the self-weights of the supernodes are on average significantly higher than the weights for transmitting states between supernodes in a level. The convergence rate can be improved by reducing the self-weights of the supernodes, so that they are on average equal to the weights between nodes. This can be done by taking advantage of the fact that

$$\left((1+\delta)P^T - \delta I\right)\pi = P^T\pi \tag{14}$$

Such an adjustment is applied to all submatrices that describe the connections between supernodes in their respective level, i.e. all block matrices on the diagonal of $\bar{\tilde{P}}$ with the exception of the first block matrix on the diagonal, which describes the connections between the base-level nodes. The δ coefficients for each level are chosen such that the mean weight for connections between nodes are equal to the mean self-weights.

Theorem 3.1: If the multilevel network with state transition matrix $\bar{\tilde{P}}$ in equation (8) has weight changes given by

$$\delta_j = \min\left\{\frac{a-b}{1-a-b}, \frac{\min\left\{\text{diag}\left(A_j\right)\right\}}{1-\min\left\{\text{diag}A_j\right\}}\right\} \tag{15}$$

$$a = \frac{1}{n}\,\text{trace}\left(A\right) \tag{16}$$

$$b = \frac{1}{n^2-n}\left\|A_j\right\|_1 - \text{trace}\left(A_j\right) \tag{17}$$

$$A_j = B_j^\dagger P B_j \tag{18}$$

then it will have the same invariant distribution as the unmodified network. With these changes, the new state transfer matrix is equation (10) in figure 3.

Proof: Adjusting the blocks on the diagonal of $\bar{\tilde{P}}$ as described above does not change the products of those entries with the corresponding parts of the invariant distribution:

$$\left((1+\delta_i)B_i^\dagger P B_i - \delta_i I\right)\gamma_i B_i^T\pi = B_i^\dagger P B_i \gamma_i B_i^T\pi \tag{19}$$

Therefore, the invariant distribution $\bar{\hat{\pi}}$ remains the same when the supernode self-weights are adjusted. ■

While adjusting super-node self-weights does not necessarily result in optimal values for $\hat{\tilde{P}}_a$, it is a heuristic that yields significant improvements in the spectral gap.

Figure 4 demonstrates the effect that adjusting supernode self-weights has on the convergence rate. For a ring-shaped network with three levels of nodes, three methods were used to construct the state transitions matrix: Metropolis weights, and the method described in the previous section with and without supernode self-weight adjustments. The computational cost of generating the networks was not taken into account here, since it is assumed that networks are used for multiple computations. Using the multi-grid method, and initial improvement in the convergence rate compared to Metropolis could be achieved, as averaging of states of nodes connected to the same supernode is accelerated compared to Metropolis weights. However, since connections between supernodes are weak, convergence slows down after a few steps. With the improvement of adjusting supernode self-weights, a significantly higher convergence rate is achieved even after these initial steps.

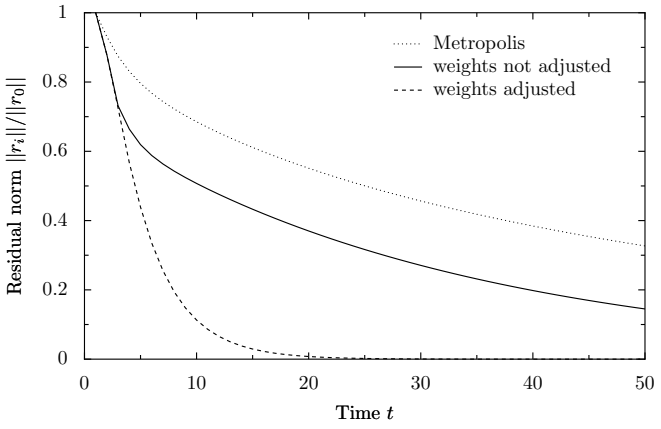

Fig. 4. Comparison of convergence for a network with three levels using Metropolis weights and the multigrid weights described here with and without supernode self-weight adjustments.

IV. Adjusting the network for broken edges and nodes

In order to be robust, the network should continue to function when one or more of its edges or nodes stop functioning, as long as the network is still connected. A broken node is a special case of multiple broken edges, since it is equivalent to breaking all edges of the effected node, and removing it from the network. One simple method for adjusting for a broken edge is for the adjacent nodes to modify their self-weights so that the row sums of the weight matrix are again equal to 1. Affected nodes only need to know the weights of their remaining edges to do this. When this method is used, the invariant distribution does not change, as long as the network is still connected. This can be shown by considering the joint probability matrix W, where

$$W_{ij} = P_{ij}\pi_i \tag{20}$$

The column sums of W are equal to the invariant distribution:

$$\sum_j W_{ji} = \pi_i \tag{21}$$

When the edge between nodes p and q is broken, P is adjusted in the following way:

$$P'_{pq} = P'_{qp} = 0 \tag{22}$$
$$P'_{pp} = P_{pp} + P_{pq} \tag{23}$$
$$P'_{qq} = P_{qq} + P_{qp} \tag{24}$$

This results in the following adjustments to W:

$$W'_{pq} = W'_{qp} = 0 \tag{25}$$
$$\frac{W'_{pp}}{\pi'_p} = \frac{W_{pp}}{\pi_p} + \frac{W_{pq}}{\pi_p} \tag{26}$$
$$\frac{W'_{qq}}{\pi'_q} = \frac{W_{qq}}{\pi_q} + \frac{W_{qp}}{\pi_q} \tag{27}$$

These adjustments preserve the symmetry of W. The column sums of W' are:

$$\sum_j W'_{ji} = \sum_j W_{ji} = \pi_i = \pi'_i \qquad \text{for } i \neq p, q \tag{28a}$$

$$\sum_j W'_{jq} = \sum_{j\neq p,q} W'_{ji} + W'_{pp} + W'_{qp} = \sum_{j\neq p,q} W_{jp} + (W_{pp} + W_{pq})\frac{\pi'_p}{\pi_p} \qquad \text{for } i = p \tag{28b}$$

$$\sum_j W'_{jq} = \sum_{j\neq p,q} W_{ji} + W'_{qq} + W'_{pq} = \sum_{j\neq p,q} W_{jq} + (W_{qq} + W_{pq})\frac{\pi'_q}{\pi_q} \qquad \text{for } i = q \tag{28c}$$

Theorem 4.1: If the multilevel network with state transition matrix $\hat{P}_a$ in equation (10) has some edges removed but remains connected, then updating the transition matrix by (22)–(24) ensures that the invariant distribution remains unchanged.

Proof: Equations (28b) and (28c) can be solved for π'_p and π'_q:

$$\pi'_p = \frac{(\pi_p - W_{pp} - W_{qp})\pi_p}{\pi_p - W_{pp} - W_{qp}} = \pi_p \tag{29}$$
$$\pi'_q = \frac{(\pi_q - W_{qq} - W_{pq})\pi_q}{\pi_q - W_{qq} - W_{pq}} = \pi_q \tag{30}$$

Therefore, $\pi'_i = \pi_i$ for all i. ■

If the network becomes disconnected as a result of broken edges, or if one or more nodes break, the resulting invariant distribution of the remaining or partial network is not the same as that of the original network, since information is lost in the process. However, the method for adjusting the network described above can still be used to determine the average of the values of the remaining nodes at the time the network was disconnected.

V. Performance and Robustness trade-offs

There are many useful measures of performance for consensus algorithms. One such performance measure is the second largest eigenvalue modulus (SLEM) [11][9]. The SLEM is a measure of the worst-case convergence rate, which applies if the initial guess is aligned with the second eigenvector, or the convergence rate that is reached when all differences in node states along other eigenvectors of the system are damped out.

Figure 5 shows the spectral gap $\rho = 1 - SLEM$ for multilevel rings with various numbers of levels. In these networks, every node is connected to its two neighboring nodes within its level, so that each level forms a ring. In addition, each node is connected to one supernode in the level above. Each supernode in one of the upper levels has the same number of subnodes.

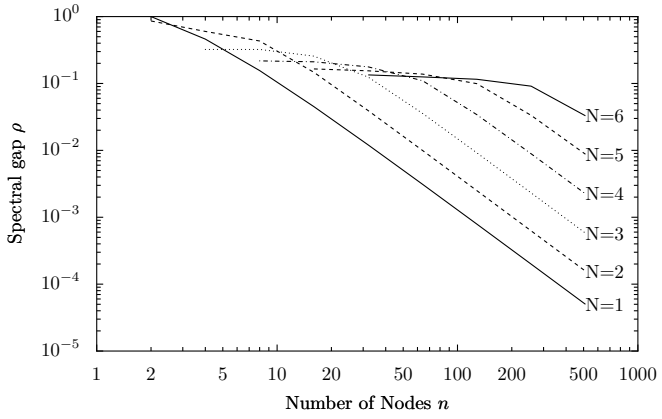

Fig. 5. Spectral gap vs. number of nodes in the base level for networks with various numbers of levels n.

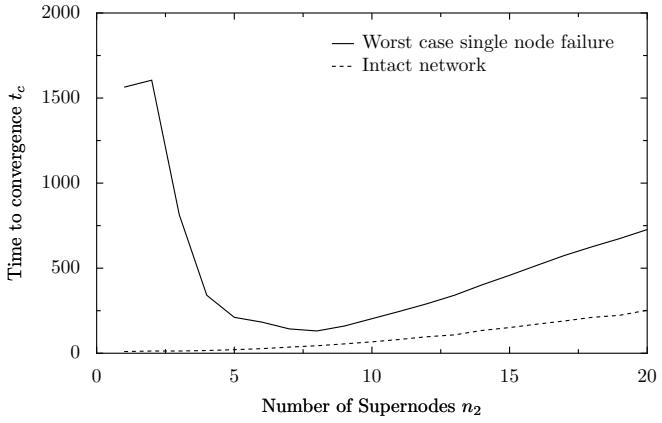

Fig. 6. Centralization Robustness vs. Performance trade-off — Single Node failure worst case performance.

As demonstrated in the figure, the spectral gap for a single-level network is inversely proportional to the square of the number of nodes of the network. However, if the number of levels in the network is sufficiently large, it scales logarithmically instead.

One simple measure of robustness is the connectivity of the network. Additional measures of robustness are necessary to evaluate how the network convergence rate is affected by failures of some edges or nodes that do not lead to parts of the network becoming disconnected. One such measures of robustness is the worst-case spectral gap of a network with a specific number of broken edges or nodes.

Another measure of performance that can be used is the inverse of the number of steps t_c required for convergence of node values to within a small error margin of the invariant distribution. Similarly, robustness can be defined as the ratio between the number of steps required for convergence for the intact network and for a network with a number of broken edges or nodes.

In constructing a multilevel network, there are a number of parameters one can chose that influence the performance and robustness of the network. The extreme cases are often

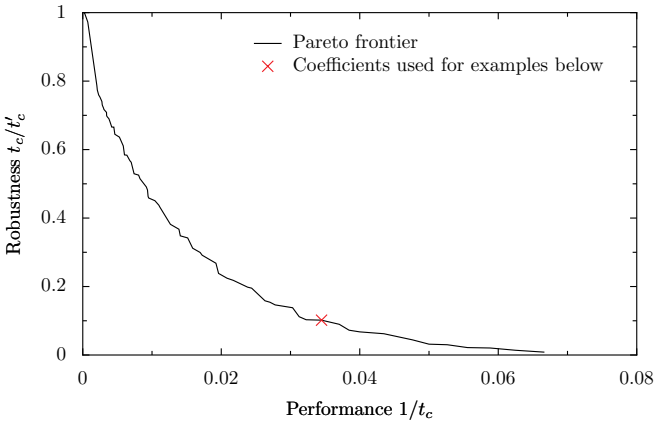

Fig. 7. Performance vs. Robustness for various α and β values.

equivalent to a single level distributed network, which is very robust but has low performance, or to a network with a single supernode, which has high performance and low robustness.

The first choices to make are the number of levels and the ratio of nodes per supernode for each level. The effects of the number of levels on the SLEM for a ring-shaped network are shown in figure 5. Figure 6 shows an example of the number of time steps required for convergence of a ring-shaped network with two levels and 40 base nodes as a function of the number of supernodes. In the case where all nodes work, the convergence rate is lower for networks with more supernodes. However, if one of the supernodes breaks, the time to convergence increases dramatically for a network with few supernodes, while networks with more supernodes are not affected as much. In this case, adding more than six supernodes to a network does not lead to faster convergence if one of them breaks, since the effect of lowering the convergence rate is larger than the benefit of added robustness. However, if several nodes malfunction, having additional supernodes can be beneficial. While the ideal number of levels depends primarily on the number of nodes in the network, the best ratio between the number of nodes in different levels depends on the expected failure rate of nodes and edges, as well as the desired level of robustness.

Additional parameters that have to be chosen are the α and β coefficients in the state transition matrix $\bar{\hat{P}}$ (figure 1). Selecting large values for the coefficients that govern data exchanges between supernodes and from supernodes to base nodes yields high performance and lower robustness, while giving base level nodes more weight increases robustness and lowers performance. Figure 7 shows the Pareto frontier of all possible combinations of these coefficients for a ring-shaped network with three levels and 64 base-level nodes. The times to convergence for the intact network and for a network with ten broken edges were used to evaluate performance and robustness. The majority of possible combinations of the four α and β parameters in this case are not on the Pareto frontier and should not be selected. Each point on the Pareto frontier represents a different performance-robustness trade-off, and

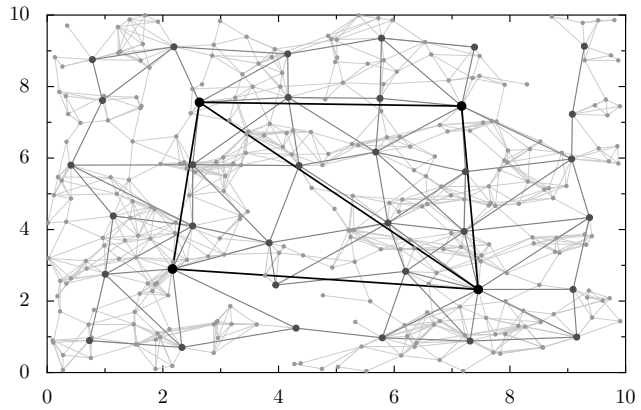

Fig. 8. Example network layout (connections between different levels are not shown).

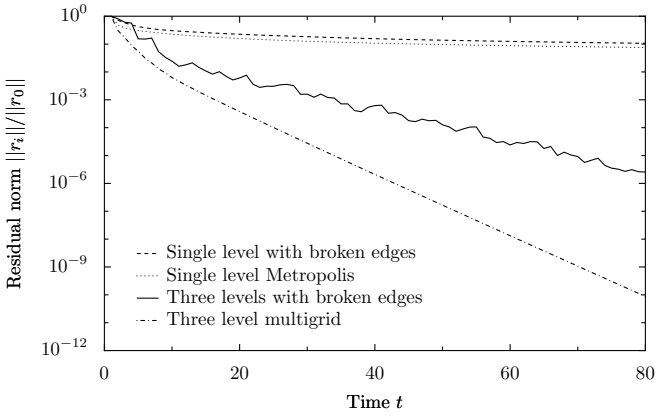

Fig. 9. Convergence results for the example network.

selection of a specific parameter combination depends on the desired level of performance or robustness.

VI. Two dimensional numerical example

To demonstrate how the algorithm described above might be used in a real network, a two dimensional network consisting of 324 randomly positioned nodes was created. The probability of having an edge between any two nodes in the base level is proportional to the square of the distance between the nodes. Two supernode levels were created by dividing the base level layout into 6×6 grid for the second level, and a 2×2 grid for the third level, and selecting the node closest to the center of each grid square to double as a supernode. The layout of this network is shown in figure 8.

Figure 9 shows the convergence of the node values to the invariant distribution for both the multigrid network and a network consisting of the base level only. As expected, the multigrid network converges significantly faster. Also plotted is a case where all edges have a probability of being functional of 0.5 at any time step. While this decreases the convergence rate, the multigrid network still performs significantly better than the single level network.

VII. Conclusion

In this paper we introduced a new multilevel and multiscale network construction that accelerates distributed consensus algorithms run on the network. We also gave update rules to show how the consensus transition matrix should be adjusted in response to node and edge failure to ensure that the invariant distribution is preserved. Using our multilevel construction we were able to explore the trade-off between heavily weighting the coarsest levels of the network, resulting in high performance but low robustness to failure, and more heavily weighting the base level, giving high robustness but low performance. The algorithms presented constitute a heuristic method. A detailed mathematical model for the resulting improvements in convergence rates would be an interesting area for future work, but is beyond the scope of this paper. The accelerated performance of consensus methods on such multilevel networks was demonstrated with an example of a random network embedded in 2D.

References

[1] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

[2] S. Boyd, A. Ghosh, and B. Prabhakar. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking*, 14:2508–2530, 2006.

[3] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri. Communication constraints in the average consensus problem. *Automatica*, 44(3):671–684, 2008.

[4] M. H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, pages 118–121, 1974.

[5] A. Ghosh and S. Boyd. Growing well-connected graphs. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 6605–6611, 2006.

[6] B. Johansson, M. Rabi, and M. Johansson. A simple peer-to-peer algorithm for distributed optimization in sensor networks. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4705–4710, 2007.

[7] J.-H. Kim, M. West, S. Lall, E. Scholte, and A. Banaszuk. Stochastic multiscale approaches to consensus problems. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 5551–5557, 2008.

[8] J.-H. Kim, M. West, E. Scholte, and S. Narayanan. Multiscale consensus for decentralized estimation and its application to building systems. In *American Control Conference, 2008*, pages 888–893, 2008.

[9] R. Montenegro and P. Tetali. *Mathematical aspects of mixing times in Markov chains*. Now Publishers Inc, 2006.

[10] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[11] J. Rosenthal. Convergence rates of Markov chains. *SIAM Rev.*, 37(3):387–405, 1995.

[12] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition, 2003.

[13] J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, 48(4):681, 2006.

[14] A. Tahbaz-Salehi and A. Jadbabaie. Small world phenomenon, rapidly mixing Markov chains, and average consensus algorithms. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 276–281, 2007.

[15] U. Trottenberg, A. Schüller, and C. W. Oosterlee. *Multigrid Methods*. Academic Press, 2000.

[16] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[17] L. Xiao, S. Boyd, and S. Lall. A space-time diffusion scheme for peer-to-peer least-squares estimation. In *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, pages 168–176, 2006.