Rewiring networks for synchronization

Aric Hagberg[1] and Daniel A. Schult[2]

[1]*Mathematical Modeling and Analysis, Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA*
[2]*Department of Mathematics, Colgate University, Hamilton, New York 13346, USA*

We study the synchronization of identical oscillators diffusively coupled through a network and examine how adding, removing, and moving single edges affects the ability of the network to synchronize. We present algorithms which use methods based on node degrees and based on spectral properties of the network Laplacian for choosing edges that most impact synchronization. We show that rewiring based on the network Laplacian eigenvectors is more effective at enabling synchronization than methods based on node degree for many standard network models. We find an algebraic relationship between the eigenstructure before and after adding an edge and describe an efficient algorithm for computing Laplacian eigenvalues and eigenvectors that uses the network or its complement depending on which is more sparse. © *2008 American Institute of Physics.*
[DOI: 10.1063/1.2975842]

Synchronization of coupled oscillators is a fundamental dynamic process with wide application in natural and technological systems.[1–4] We consider generic oscillators with diffusive-type coupling through a network and examine how the network structure affects synchronization. Network structure measures have developed at a fast pace in the last 10 years[5] and many statistical measures of network structure such as degree distribution, clustering and degree associativity are commonly used when describing networks. But how does network structure relate to synchronization? In this paper, we consider rewiring networks as a local view of network structure's impact on synchronization. We model the synchronization of identical oscillators via a master stability function approach[6–8] where the topology of the network, as measured by the Laplacian matrix eigenvalues, is separated from consideration of the dynamics of the oscillators, as measured by the master stability function. This separation means that a given network is more apt to allow synchronization regardless of the type of oscillator. The synchronized solution is linearly stable if the network Laplacian spectrum lies in an interval where the master stability function is negative. Our goal in this paper is to examine how to engineer a network to enhance synchronization. We determine the impact on synchronization of rewiring (single edge manipulations). This leads to strategies for choosing edges that most enhance synchronization.

I. INTRODUCTION

Synchronization in networks of coupled oscillators is a fundamental problem with applications in ecosystem dynamics,[9] communication using chaos,[10] muscle tissue such as the heart,[1] and mitochondrial membrane potentials.[11] In addition, synchronization of neuron networks is critical to understanding such wide ranging brain functions as long range communication, effective modulation of motor neuron input, and stabilization of response under variable levels of neurotransmitter.[12] Some theories of neural information encoding rely on synchronization as the mechanism of encoding stimulus and higher processing of that stimulus.[13] Many studies have attempted to relate network structural properties to synchronization. A common intuition is that networks with small diameter or small average path length should be easier to synchronize.[6,14–16] But this intuition is not correct.[14,16,17] Careful analysis of the probability distribution of eigenvalues in the limit of large networks shows that other relationships do hold between statistical measures of network structure and synchronization properties.[18,19] Despite this, exceptional cases exist for many statistical constraints.[17] Even small changes in network structure, such as adding or removing an edge[20–22] or changing edge weights[23] can affect the ability of the network to synchronize in ways that are not fully understood.

Attempts to design networks with specific dynamic properties can be grouped by design constraints: constructing networks with generative models,[24–26] changing the edge weights, coupling strength, or directionality,[27,28] or by rewiring an existing network through removal,[22] addition, or moving of single edges, or swapping pairs of edges to maintain a specific degree sequence.[29] In this paper, we focus on the effects of adding and removing edges and especially the combination of both as single edge rewiring operations.

The network Laplacian arises naturally in the study of coupled oscillators and the Laplacian spectrum plays a key role in determining synchronization properties.[2,6] Analyzing the eigenvalues has provided insights into optimizing the synchronization of random,[25] evolving,[30] and directed weighted networks.[27,31,32] The Laplacian eigenvector structure also provides information relevant to synchronization, for example, by defining oscillation modes which become unstable as a system is desynchronized when decreasing the coupling strength[33] or in determining methods for course graining large systems.[34] Partial synchronization occurs when only some of the modes are unstable and those eigen-

vectors describe spatial properties of the dynamics well beyond the stability boundary.[33] Some oscillator systems linearize to equations involving the adjacency matrix of a network. In that setting the eigenvectors of the adjacency matrix have been used to study the synchronization of networks when nodes or edges are removed.[22] Here, we instead focus on diffusively coupled systems which upon linearization involve the Laplacian matrix. We exploit the eigenvectors for the modes of the extremal eigenvalues to design algorithms which optimally add or remove edges with the goal of expanding the range of coupling strength for which the synchronized solution is linearly stable. We point out that these algorithms can also be inverted to inhibit synchronization though we focus on enhancing it.

We start by considering a system of n identical oscillators symmetrically coupled through a network. The equations of motion for the oscillator state vector x_i at each node i are

$$\frac{dx_i}{dt} = F(x_i) - \sigma \sum_{j=1}^{n} L_{ij} H(x_j), \tag{1}$$

where $F(x)$ determines the uncoupled oscillator dynamics of each node, $H(x)$ specifies the coupling of the vector fields, and σ is the coupling strength. The topology of the network is specified through the coefficients L_{ij} of the Laplacian matrix. The diagonal elements L_{ii} are the degree of node i. Off-diagonal elements L_{ij} take the value -1 if an edge exists between node i and node j and 0 otherwise.[35]

Solutions of Eq. (1) are defined to be synchronized if $x_i(t) = x_j(t)$ for all nodes i and j in the network. The system is said to be synchronizable if a synchronized solution is linearly stable to nonuniform perturbations. To determine whether the system is synchronizable we follow the approach in Ref. 6 and linearize Eq. (1) about the synchronized solution $x_i(t) = s(t)$ for all i. The linearized equations can be diagonalized into n blocks, indexed by i, of the form

$$\frac{dy_i}{dt} = [J_F(s) + \sigma \lambda_i J_H(s)] y_i, \tag{2}$$

where J_F and J_H are the Jacobians of F and H, λ_i is an eigenvalue of L, and y_i represents the displacement from the synchronized solution. The stability of the synchronized state is determined by the largest Lyapunov exponent of the block system given by the master stability function $\Gamma(\sigma\lambda)$ for Eq. (2).[6–8] If $\Gamma(\sigma\lambda_i) < 0$ for each $i \geq 2$, the synchronized state is linearly stable.[36] For many oscillatory systems the master equation is negative only in a single interval $[\alpha_1, \alpha_2]$ determined by F, H, and s and not by the network topology. This implies that the network is synchronizable only when the ratio $r \equiv \lambda_n / \lambda_2 < \alpha_2 / \alpha_1$,[6] and that once the dynamics of the individual oscillators are specified the synchronization problem reduces to studying the spectrum λ_i of the Laplacian.

For some special networks such as threshold networks, the spectrum is known exactly and r can be computed easily.[24] Other networks such as those constructed using Cartesian product, join, and coalescence provide a way to compute the spectrum during construction.[20] For networks where the spectrum is not known exactly, bounds on the Laplacian spectrum[20,37–41] give estimates on the eigenratio r. The maximal eigenvalue λ_n is bounded between the largest degree k_n and the size of the network, $k_n + 1 \leq \lambda_n \leq n$, and the first non-zero eigenvalue λ_2, often called the algebraic connectivity, or spectral gap, is bounded above by the lowest degree in the network $\lambda_2 \leq k_1$ (if the network is not a complete graph). Lower bounds on λ_2 exist, such as $2[1 - \cos(\pi/n)]$ and $4/nd$, where d is the diameter of the network, but they are not tight bounds for most networks. Putting these bounds together for the eigenratio r we find

$$\frac{k_n + 1}{k_1} \leq r \leq \min\left\{ \frac{n}{2[1 - \cos(\pi/n)]}, \frac{n^2 d}{4} \right\}. \tag{3}$$

If the spectrum cannot be accurately estimated or analyzed using these techniques the eigenvalues must be computed.

The algebraic connectivity λ_2 also appears in the study of the topological structure of graphs. In that context λ_2 is a measure of how well connected a graph is; a graph with the same node set and larger algebraic connectivity is considered more connected. Rewiring graphs for larger algebraic connectivity is directly related to the case we discuss below of adding edges to reduce r and a performance analysis for growing small graphs with a greedy heuristic is given in Ref. 42.

II. LAPLACIAN EIGENVECTORS

In addition to the eigenvalues, the eigenvectors v_k are useful and provide information about how the eigenvalues change when edges are added and removed. Each eigenvector v_k associates to node i a value $v_{k,i}$. In the following we show how to use these values to choose edges which most impact synchronization.

Consider the Laplacian of a given network after adding an edge $\tilde{L} = L + L_e$, where L_e is the Laplacian for the network consisting of a single edge. The characteristic polynomial is

$$\det(L + L_e - \lambda I) = 0, \tag{4}$$

which when transformed to the coordinate system given by the eigenvectors of L yields

$$\det(\Lambda + \Delta^T \Delta - \lambda I) = 0, \tag{5}$$

where Λ is the diagonal matrix of (sorted) eigenvalues of L and Δ is the vector with elements $\Delta_k = |v_{k,i} - v_{k,j}|$, where edge (i,j) is being added. The term $\Delta^T \Delta$ in Eq. (5) is an outer product and thus has rank 1. Noting that the first coordinate Δ_1 is zero for any edge, a general form for $\tilde{L}$ is

$$\tilde{L} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 + \Delta_2^2 & \Delta_2 \Delta_3 & \cdots & \Delta_2 \Delta_n \\ 0 & \Delta_3 \Delta_2 & \lambda_3 + \Delta_3^2 & \cdots & \Delta_3 \Delta_n \\ 0 & \vdots & \vdots & \ddots & \vdots \\ 0 & \Delta_n \Delta_2 & \cdots & \cdots & \lambda_n + \Delta_n^2 \end{bmatrix}. \tag{6}$$

If only one eigenvector v_k differs across the added edge, then only one eigenvalue changes $\tilde{\lambda}_k = \lambda_k + \Delta_k^2$ and all other eigenvalues remain the same. If only two eigenvectors, say v_k and v_ℓ differ across the added edge, the two modified eigenvalues are given by

$$\tilde{\lambda} = \frac{Q_k + Q_\ell}{2} \pm \sqrt{\left(\frac{Q_k - Q_\ell}{2}\right)^2 + \Delta_k^2 \Delta_\ell^2}, \qquad (7)$$

where $Q_i \equiv \lambda_i + \Delta_i^2$. If m eigenvectors differ across an added edge, then we must solve for the eigenvalues of an $m \times m$ matrix. This is potentially much faster than solving the original $n \times n$ problem. Similar computations work for removing an edge, where L_e is subtracted instead of added.

III. ANALYSIS OF EIGENVECTORS FOR ADDING OR REMOVING EDGES

We now discuss strategies for selecting which edge to add or remove. To reduce the eigenratio r we want to decrease λ_n (removing edges can never increase λ_2, Ref. 37). By definition the largest Laplacian eigenvector, v_n satisfies

$$\lambda_n = \sum_{i \sim j} (v_{n,i} - v_{n,j})^2 \quad \text{subject to} \quad \sum_i v_{n,i}^2 = 1. \qquad (8)$$

This suggests that we remove the edge (i,j) which maximizes $|v_{n,i} - v_{n,j}|$. This choice of edge is optimal in the following perturbation sense. We can track changes in the eigenvalues upon removing an edge by analyzing $L - sL_e$ as a weight s varies from 0 to 1. From Eq. (8) we see this perturbation subtracts the term $s^2(v_{n,i} - v_{n,j})^2$ from the sum of squares, and also changes the sum due to the normalization constraint. Since, as s increases from 0, the change in λ_n due to normalization is zero (λ_n is maximal in these coordinate variables when $s=0$), we obtain

$$\frac{d\lambda_n}{d(s^2)} = -(v_{n,i} - v_{n,j})^2. \qquad (9)$$

The choice of edge with most disparate nodes in v_n is optimal for infinitesimal edge removal. For full edge addition or removal, this argument for optimality does not hold because the normalization constraint can counteract the gains due to the new term. But, these eigenvector based methods remain an effective heuristic technique for selecting edges.

The same ideas work for the lowest nonzero eigenvalue and associated eigenvector (λ_2, v_2).[42] There are two impacts of adding or removing an edge. One is due to the addition or removal of a term in the sum and the other is due to the normalization constraint changing entries in v_2 and thus the other terms in the sum. In the case of λ_2, we can add the edge with most disparate nodes in v_2 and this choice is optimal for infinitesimal edge addition.

In searching for edges most disparate in the eigenvector, we do not need to examine every edge in the network. Rather we can examine the values in v_n and determine which two nodes are most disparate (see Fig. 1). Often an edge exists between them and can be removed. In our experience, this is the case in the majority of networks considered. In some circumstances though, those nodes are not connected with an edge. We can construct such an example network starting with a ring network for which v_n alternates sign at each node around the ring. Symmetry ensures that any edge is as good as another, but we can break the symmetry by connecting a special node to its four closest nodes on the ring. If we embellish the network in this way with two special nodes (not
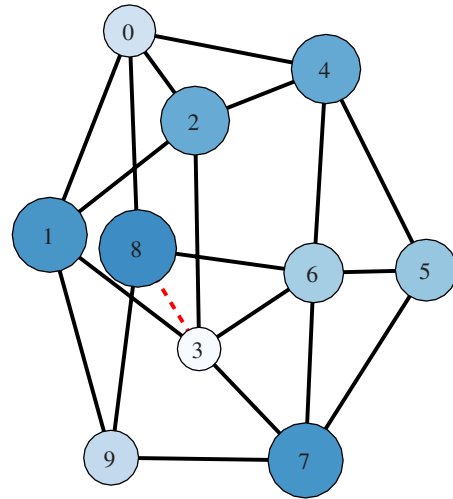

FIG. 1. (Color online) Choosing an edge to remove based on the eigenvector v_n of the largest eigenvalue λ_n for a small network. The node sizes and colors are proportional to the value of $v_{n,i}$ at each node i. The edge (3,8), shown by the dashed line, has the largest value of $|v_{n,i} - v_{n,j}|$ for all edges (i,j) and is chosen to be removed. In this graph nodes 3 and 6 have the highest degree (5) but disconnecting the edge between them is not the optimal choice. Upon removal of the edge (3,8) the resistance r decreases from 3.85 to 3.67. Similar treatment for adding edges leads to examination of the eigenvector v_2. Combining these two approaches allows us to move edges as well thus keeping the overall number of edges fixed.

directly across from each other nor close enough to overlap neighborhoods) those special nodes will be most disparate in the eigenvector, yet they are not connected. In cases such as this, we still do not need to consider each edge to find the optimal one. It is more effective to sort nodes by eigenvector value and consider pairing the high and low values in decreasing difference order until an edge exists.

Using spectral information to select edges with maximal impact on synchronization has also been successfully used for coupled oscillators in a directed network using the adjacency matrix and its spectrum. In Ref. 22 a quantity called the dynamical importance is introduced to measure the change in eigenvalue when removing an edge. This quantity is estimated using a perturbation argument and is used to propose a choice of edge which maximizes the dynamical importance. The perturbation argument can be adapted to the Laplacian matrix setting and leads to an edge removal strategy that is equivalent to ours. One difference between strategies for directed adjacency coupling and undirected Laplacian coupling is that in the latter we can search for the most disparate nodes in the eigenvector instead of searching all edges for the edge with most dynamical importance. In addition our derivation allows us to study adding edges using the eigenvector with the lowest nonzero eigenvalue and moving edges by combining both adding and removing edges.

IV. GREEDY ALGORITHMS FOR ADDING, REMOVING, OR MOVING EDGES

We tested the effectiveness of spectral edge removal strategies by implementing a greedy algorithm that repeatedly removes edges while re-evaluating the removal criteria at each step. We compared this eigenvector approach with
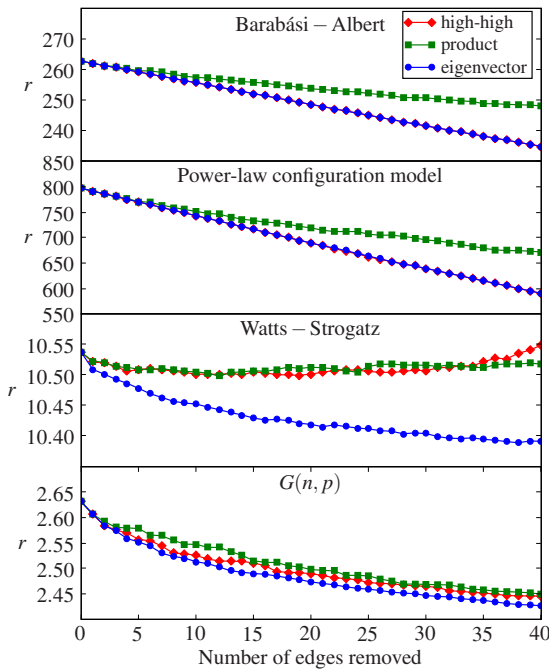
FIG. 2. (Color online) Three methods for choosing two nodes to disconnect: the highest degree node and its highest degree neighbor (high-high); the highest product of degrees connected by an edge (product); the nodes with most disparate values in the eigenvector v_n (eigenvector). Disconnecting the two highest degree nodes is as effective in reducing r as the eigenvector method for network models with a skew degree distribution (Barabási–Albert, power-law configuration model). But over all models the eigenvector method is the most effective.
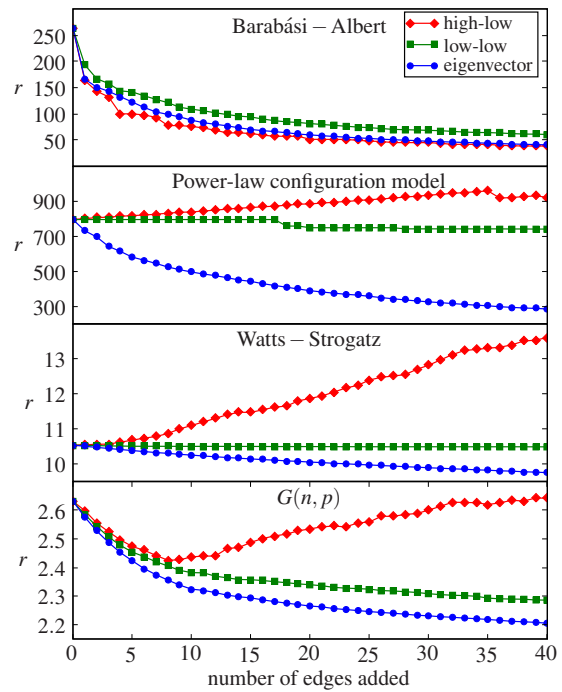

FIG. 3. (Color online) Three methods for choosing two nodes to connect: the highest and lowest degree nodes (high-low); the two lowest degree nodes (low-low); the nodes with most disparate values in the eigenvector v_2 (eigenvector). For adding edges the eigenvector algorithm is clearly most effective except in the case of the Barabási–Albert model. In that model the eigenvector v_2 singles out a low degree node and lumps the other nodes together. Thus a hub node with high degree and v_2 value close to most disparate causes the effect of eigenvector normalization in Eq. (8) to dominate the impact of connecting the hub to the singled out node. When the normalization constraint dominates in this way the eigenvector strategy is slightly less effective.

two other edge selection strategies based on degree: disconnecting the node with the highest degree from its highest degree neighbor ("high-high"), and removing the edge with the largest product of degrees for the nodes at each end ("product").[21]

For our tests we used four different network models:[5] Barabási–Albert preferential attachment with $m=30$; configuration model with a power-law degree distribution $P(k) \sim k^{-2}$; Watts–Strogatz small world with $k=20$, $p=0.1$; and Erdős–Rényi $G(n,p)$ random graph with $p=0.1$. While we varied the network density from 0.01 to 0.3 and number of nodes from 20 to 2000 in our explorations, these variations change the results quantitatively but not qualitatively, so here we present one representative case except where noted otherwise. For each model we used four realizations of $n=500$ nodes. The parameters in the network models were chosen to produce networks with a density of approximately 0.1 (except for power-law configuration model which has a density of about 0.01). If the model produced a disconnected network [e.g., configuration model and $G(n,p)$] we used the largest connected component so in that case $n \approx 500$. Since a connected graph is required for synchronization, in all strategies we do not permit removing an edge that would disconnect the graph.

Figure 2 shows that the eigenvector method performs the best over our set of sample networks. We note here that for all network models, random edge removal has no average effect on r and thus we do not show those results. For the Barabási–Albert, Erdős–Rényi, and configuration model net-

works, the "high-high" method and eigenvector method perform almost exactly the same indicating that high degree nodes are associated with large eigenvalues. There is a noticeable difference for the Watts–Strogatz model network, where the node degrees are almost all the same, which shows that the eigenvectors encode more information than just node degree.

Another rewiring approach is to add targeted edges to reduce r. Since adding edges always increases eigenvalues[37] we strategically choose edges which impact λ_2.[42] Our eigenvector method examines v_2 and adds the edge (i,j) that maximizes $|v_{2,i}-v_{2,j}|$ [subject to the edge (i,j) not already in the network]. This approach is optimal in the same perturbation sense discussed above. We compare this eigenvector method with two degree oriented approaches: connecting the lowest degree node to the second lowest degree node ("low-low") and connecting the highest degree node to the lowest degree node ("high-low").

Figure 3 shows that the eigenvector method performs better than the node-degree methods over our sample set of network models with one exception. For the Barabási–Albert (BA) model, the "high-low" method performs slightly better even though it is not at all effective on the other sample networks. We discuss why this might be below. The figure also shows that adding edges which impact λ_2 is more effective in lowering r than removing edges which impact λ_n.
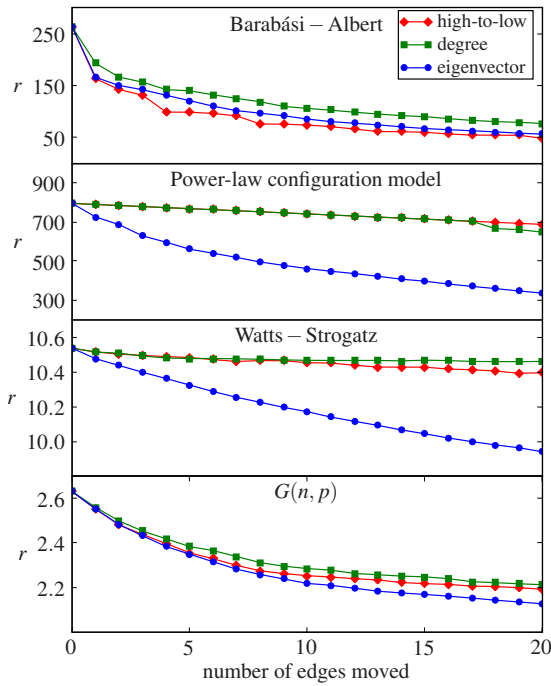
FIG. 4. (Color online) Three methods for moving (adding and removing) an edge. The first removes the edge between nodes most disparate in the eigenvector v_n and adds an edge between nodes most disparate in v_2 (eigenvector). The other two methods remove the edge between the highest degree node and its highest degree neighbor and then add either an edge between the highest and lowest degree nodes (high-to-low) or an edge between the two lowest degree nodes (degree). In comparing to other figures, 20 edges moved is the same number of operations as 40 edges added (or removed). For the combination of adding and removing an edge, the eigenvector strategy performs the best except for the Barabási–Albert model. In that case the high-to-low method performs better because adding edges connecting low degree nodes to hubs causes normalization constraints to dominate the change in the spectrum as described in Fig. 3.

This is reasonable because the sum of the entire spectrum decreases (increases) by 2 when an edge is deleted (added). This fixed absolute change affects the lower λ_2 at a much higher relative rate than λ_n.

A third rewiring approach is to move an edge. For the eigenvector method we combine the methods from the above based on the eigenvectors v_2 (adding) and v_n (removing). In Fig. 4 we compare this method with two heuristic methods based on node degrees. The "degree" method uses the previous methods of connecting low degree nodes to low degree neighbors and disconnecting high degree nodes from high degree neighbors. The "high-to-low" method first disconnects the high degree nodes and then connects the lowest degree node to the highest degree node. Our results show that the eigenvector method again has the best performance on our set of sample networks. As for adding edges, some special networks allow degree based methods to perform slightly better though they perform much worse on other networks. We also find that moving edges are about as effective as adding edges for reducing r and is a useful strategy if the total number of edges in the network should be kept constant.

To understand why the "high-low" method performs better than the eigenvector method for some networks, we consider the two effects that adding an edge has on λ_2 in Eq. (8):
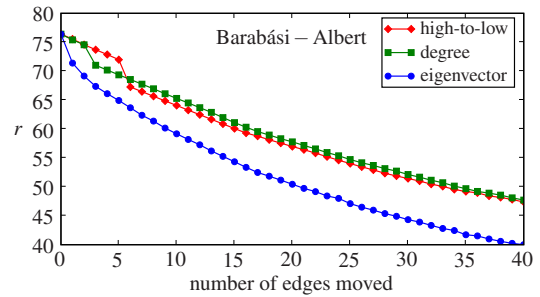

FIG. 5. (Color online) The affect of moving edges on resistance is shown for Barabási–Albert model networks with a lower density. Here $m=3$ with a density of 0.006 (instead of $m=30$ with a density of 0.1). Comparing the top portion of Fig. 4 we see that reducing the density with heterogeneous networks allows the eigenvector method to again perform better than the degree-based methods.

(1) the addition of a new term and (2) changes in the other terms due to the normalization constraint. For infinitesimal edge addition ($s \ll 1$) the sum increases exclusively due to the new term being added. But as $s \rightarrow 1$ to represent the addition of the full edge, changes in the sum also occur due to the normalization constraint. In some networks, e.g., for the Barabási–Albert model, the eigenvector v_2 is lopsided such that one low-degree node i has a large value $v_{2,i}$ of one sign and the remaining nodes all have small values of v_2 of the opposite sign. (The sum of the components is zero to be orthogonal to the ever present constant eigenvector.) The eigenvector method connects node i to the node j with the largest $|v_{2,i} - v_{2,j}|$ which is often a low-degree node "on the other side" of a hub even though there exist high degree nodes with values near $v_{2,j}$. When node j has low degree it does not appear in many of the terms in Eq. (8) and so changes due to normalization are small. The "high-low" method, on the other hand, connects node i to a high degree node which affects a large number of terms in Eq. (8). In this case, the impact due to normalization constraints can make up for the slightly less optimal choice of a new term to add.

In Fig. 5 we show that for a less dense version of the Barabási–Albert model with $m=3$ the eigenvector method is again significantly better than the "high-to-low" method. The lower density means that even high degree nodes impact fewer terms in Eq. (8) so that introducing a new term dominates the changes due to normalization constraints. The eigenvector heuristic is based on introducing the largest possible new term so it does not perform as well when large degree hubs take on near-extremal values in the eigenvector. Thus in general, methods which take advantage of high degree nodes may in special circumstances do better than the eigenvector method when the effect of normalization constraints on the eigenvalue are greater than the cost of adding a smaller new term to the sum. The only circumstances in which we have seen this are for heterogeneous and yet relatively dense networks.

Because moving edges leaves the number of edges the same, we can continue to move edges until the process converges. In Fig. 6 we show the resistance level over 1200 edge moves using the eigenvector method for network types with approximately the same number of nodes and edges. The
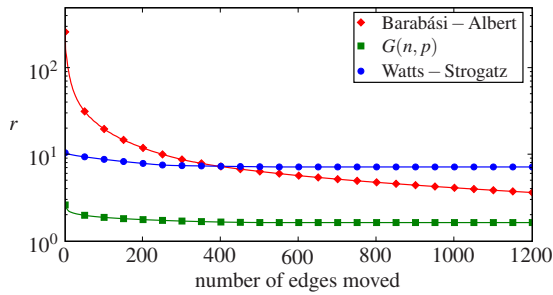
FIG. 6. (Color online) The convergence of moving edges using the eigenvector method is shown for each network type by plotting 1200 edge moves. It is apparent that convergence occurs and that the resulting resistance levels differ by network type. Clearly the greedy strategy is effective at reducing r for each network type but may not achieve a global minimum for r. The power-law configuration model results are similar but not shown because of their potentially misleading lower density. The graphs shown are based on networks that have approximately the same density: 500 nodes and 12000 edges.

resistance levels appear to converge with resulting resistance levels that differ by network type suggesting that the minimum found depends on the network type. Said another way, when considering networks with fixed number of nodes and edges the resistance r has local minima which are not the global minimum. In general, moving individual edges with our "greedy" heuristic algorithm will often find a local minimum which depends on the initial network structure.

V. COMPUTING EXTREMAL EIGENVALUES

We now turn to some comments about computing λ_2, λ_n and the corresponding eigenvectors. For large networks, finding these eigenvalues and eigenvectors can be computationally intensive. For the common case of large sparse matrices, iterative techniques such as the power method and related methods are effective.[43] For the special case of the network Laplacian we can use the Laplacian of the complement network to provide further improvements. The complement of a network places an edge between two nodes precisely when there is not an edge in the original network. The Laplacian of the complement network is related to the original network by

$$L^C = nI - \mathbf{1} - L = (nI - D) - \mathbf{1} + A, \tag{10}$$

where n is the number of nodes in the network, I is the identity matrix, D is the diagonal degree matrix, A is the adjacency matrix, and $\mathbf{1}$ is the matrix of all ones. The eigenvalues of L^C are $\lambda_1^C = 0$, $\lambda_i^C = n - \lambda_i$, $i = 2, \ldots, n$ and L^C and L share the same set of eigenvectors.[37,38]

To see how Eq. (10) is used in computing λ_2 and λ_n, suppose we have a sparse network. We use power iteration to find λ_n through repeated (sparse matrix) multiplication of a trial vector v by L. To find λ_2 we note that v_2 is also the eigenvector of L^C with the largest eigenvalue $n - \lambda_2$ so we could use power iteration with L^C to find λ_2. But, since L^C is not sparse, we can instead use Eq. (10) to keep the advantage of sparse matrix multiplication. If we keep our second trial vector w orthogonal to the constant vector by subtracting its average value, we ensure that multiplying by $\mathbf{1}$ results in the zero vector. Then the power method iterative step for the trial

vector w is $w_{\ell+1} = (nI - D + A)w_\ell$ followed by subtraction of its average. This allows the computation to remain sparse at the small cost of ensuring orthogonality to the constant vector. In practice, shifting the vector elements to be orthogonal to $\mathbf{1}$ is not needed at every iteration so further performance tuning is possible. Since the matrices A and D are already used in the power method iteration of L in computing λ_n, we see that λ_2 requires approximately the same computation as finding λ_n.

The convergence rate of the power method is determined by the ratio of the magnitude of largest two eigenvalues λ_{n-1}/λ_n [similarly for $(n - \lambda_3)/(n - \lambda_2)$]. The practical usefulness of power iteration depends on this ratio which may be one or close to one for some networks. In that case variants of the Krylov subspace iterative methods may be more effective in finding the extreme eigenvalues.[43] Taking advantage of the complement network to ensure sparse matrix consideration is still effective.

VI. CONCLUSIONS

In summary we have explored the impact of adding, removing, and moving single edges on the synchronization of coupled oscillator networks. We described how to compute the relationship between the eigenstructure before and after adding or removing an edge. The problem of finding the eigenvalues in the rewired network requires solving an $m \times m$ eigensystem where m is the number of eigenvectors in the original network which differ across the edge to be added or removed.

We compared various rewiring schemes designed to increase the ability of a network to synchronize. We find that while all strategic rewiring methods are much more effective than random rewiring, schemes based on the Laplacian eigenvectors of the network are usually more effective than computationally cheaper schemes based on targeting the nodes of largest and smallest degree. Though we have considered networks without edge weights, the same technique can be applied to weighted networks.

For the methods we studied, adding edges selected to impact λ_2 always enhanced synchronization better than removing edges selected to impact λ_n. Moving edges has the advantage of maintaining the number of edges in the network but does not appear to be better at enhancing synchronization.

For some problems in ecosystem dynamics[9] and neuron networks,[12] the goal is to inhibit synchronization. In those cases our strategies may be inverted. We remove edges selected to impact λ_2 and/or add edges selected to impact λ_n. Eigenvector methods are still effective and removing edges becomes better than adding because the relative impact of an edge on λ_2 is larger than the impact on λ_n.

ACKNOWLEDGMENTS

[1] S. H. Strogatz, *Sync: The Emerging Science of Spontaneous Order* (Hyperion, New York, 2003).
[2] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Phys. Rep. **424**, 175 (2006).
[3] J. A. Acebrón, L. L. Bonilla, P. C. J. Vicente, F. Ritort, and R. Spigler, Rev. Mod. Phys. **77**, 137 (2005).
[4] S. H. Strogatz, Physica D **143**, 1 (2000).
[5] M. E. J. Newman, SIAM Rev. **45**, 167 (2003).
[6] M. Barahona and L. M. Pecora, Phys. Rev. Lett. **89**, 054101 (2002).
[7] L. M. Pecora and T. L. Carroll, Phys. Rev. Lett. **80**, 2109 (1998).
[8] K. S. Fink, G. Johnson, T. Carroll, D. Mar, and L. Pecora, Phys. Rev. E **61**, 5080 (2000).
[9] D. J. D. Earn, S. A. Levin, and P. Rohani, Science **290**, 1360 (2000).
[10] L. M. Pecora, T. L. Carroll, G. A. Johnson, D. J. Mar, and J. F. Heagy, Chaos **7**, 520 (1997).
[11] M. A. Aon, S. C. Cortassa, and B. O'Rourke, Biophys. J. **91**, 4317 (2006).
[12] A. Schnitzler and J. Gross, Nat. Rev. Neurosci. **6**, 285 (2005).
[13] W. Singer, Neuron **24**, 49 (1999).
[14] T. Nishikawa, A. E. Motter, Y. C. Lai, and F. C. Hoppensteadt, Phys. Rev. Lett. **91**, 014101 (2003).
[15] F. M. Atay, T. Bıyıkoğlu, and J. Jost, IEEE Trans. Circuits Syst., I: Regul. Pap. **53**, 92 (2006).
[16] A. E. Motter, C. Zhou, and J. Kurths, Phys. Rev. E **71**, 016116 (2005).
[17] F. M. Atay, T. Bıyıkoğlu, and J. Jost, Physica D **224**, 35 (2006).
[18] D.-H. Kim and A. E. Motter, Phys. Rev. Lett. **98**, 248701 (2007).
[19] S. Guan, X. Wang, K. Li, B.-H. Wang, and C.-H. Lai, Chaos **18**, 013120 (2008).
[20] F. M. Atay and T. Bıyıkoğlu, Phys. Rev. E **72**, 016217 (2005).
[21] C. Y. Yin, W. X. Wang, G. Chen, and B. H. Wang, Phys. Rev. E **74**, 047102 (2006).
[22] J. G. Restrepo, E. Ott, and B. R. Hunt, Phys. Rev. Lett. **97**, 094102 (2006).
[23] H. Sorrentino, G. di Bernardo, S. Huerta Cuéllar, and S. Boccaletti, Physica D **224**, 123 (2006).
[24] A. Hagberg, P. J. Swart, and D. A. Schult, Phys. Rev. E **74**, 056116 (2006).
[25] L. Donetti, P. I. Hurtado, and M. A. Muñoz, Phys. Rev. Lett. **95**, 188701 (2005).
[26] R. M. Memmesheimer and M. Timme, Physica D **224**, 182 (2006).
[27] T. Nishikawa and A. E. Motter, Physica D **224**, 77 (2006).
[28] A. E. Motter, New J. Phys. **9**, 182 (2007).
[29] B. Wang, T. Zhou, Z. L. Xiu, and B. J. Kim, Eur. Phys. J. B **60**, 89 (2007).
[30] F. Sorrentino and E. Ott, Phys. Rev. Lett. **100**, 114101 (2008).
[31] T. Nishikawa and A. E. Motter, Phys. Rev. E **73**, 065106 (2006).
[32] J. G. Restrepo, E. Ott, and B. R. Hunt, Chaos **16**, 015107 (2006).
[33] P. N. McGraw and M. Menzinger, Phys. Rev. E **75**, 027104 (2007).
[34] D. Gfeller and P. D. L. Rios, Phys. Rev. Lett. **100**, 174104 (2008).
[35] The Laplacian $L = D - A$, where D is a diagonal degree matrix and A is the adjacency matrix for the network. Alternatively the normalized Laplacian $\mathcal{L} = I - D^{-1/2} L D^{-1/2}$ is sometimes used. In both cases, networks with weighted edges can be considered. Appropriate adjustments to our analysis work for these models of connectivity and produce similar results.
[36] We denote the sorted Laplacian eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and corresponding eigenvectors $v_1, v_2, \ldots v_N$. The eigenvalue $\lambda_1 = 0$ corresponds to spatially uniform perturbations $v_1 = [1, \ldots, 1]$.
[37] R. Merris, Linear Algebr. Appl. **198**, 143 (1994).
[38] R. Merris, Linear Algebr. Appl. **278**, 221 (1998).
[39] R. Grone, R. Merris, and V. S. Sunder, SIAM J. Math. Anal. **11**, 218 (1990).
[40] B. Mohar, *Graph Theory, Combinatorics, and Applications*, edited by Y. Alavi, G. Chartrand, O. R. Oellermann, and A. J. Schwenk (Wiley, New York, 1991), Vol. 2, pp. 871.
[41] F. R. K. Chung, "Spectral Graph Theory," Vol. 92 in *Regional Conference Series in Mathematics* (American Mathematical Society, New York, 1997).
[42] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *Proceedings of the 45th IEEE Conference on Decision and Control* (IEEE, San Diego, CA, 2006), pp. 6605–6611.
[43] Y. Saad, *Numerical Methods for Large Eigenvalue Problems* (Manchester University Press, Manchester, UK, 1992).