Decentralized Estimation and Control of Graph Connectivity for Mobile Sensor Networks

Peng Yang Randy A. Freeman Geoffrey J. Gordon Kevin M. Lynch Siddhartha S. Srinivasa
Rahul Sukthankar

***Abstract*— The ability of a robot team to reconfigure itself is useful in many applications: for metamorphic robots to change shape, for swarm motion towards a goal, for biological systems to avoid predators, or for mobile buoys to clean up oil spills. In many situations, auxiliary constraints, such as connectivity between team members or limits on the maximum hop-count, must be satisfied during reconfiguration. In this paper, we show that both the estimation and control of the graph connectivity can be accomplished in a decentralized manner. We describe a decentralized estimation procedure that allows each agent to track the algebraic connectivity of a time-varying graph. Based on this estimator, we further propose a decentralized gradient controller for each agent to maintain global connectivity during motion.**

I. Introduction

A *mobile sensor network* consists of n mobile sensors (or agents) connected by links along which information flows. Applications for mobile sensor networks include target tracking [15], [25], [22], [12], formation and coverage control [1], [2], [4], [6], environmental monitoring [10], [11], [17], [20], and several others. These applications take advantage of the sensors' ability to position themselves to maximize the information in their sensor readings. For these cooperative sensing applications, it is often desirable to maintain a connected communication graph, even as communication links are established or lost as the agents move. To date, the connectivity-maintenance problem has been addressed using two different approaches: control of local connectivity measures using decentralized control schemes, and control of global connectivity measures based on centralized computations.

The first approach focuses on devising decentralized controllers for each agent to maintain local connectivity. For discrete-time second-order agents, a feasible control space is computed in [14] for each agent to maintain all existing pairwise connections. In comparison, in [18] each agent tries to maintain its two-hop communication neighbors. The use of local connectivity measures allows each agent to compute a feasible motion controller with only local information. In many cases, however, it is the global connectivity of the network that is of primary interest, and strict maintenance of local connectivity may be overly restrictive.

The second approach in [3], [23], [24] uses global connectivity measures such as *algebraic connectivity* [5]. Given a graph, a k*-connectivity matrix*[1] is computed in [23]. To maintain graph connectivity, gradient controllers are designed such that each off-diagonal entry of the k-connectivity matrix, where $k = n$, remains positive over time. In comparison, the gradient controller designed in [24] uses the fact that connectedness of the graph is equivalent to the determinant of the *deflated Laplacian matrix* being positive. However, computing the k-connectivity matrix and the determinant of a deflated Laplacian matrix are both centralized procedures. A method to compute the k leading eigenvectors of an n by n matrix is proposed in [9]. In [3] this method is used to compute *Fiedler eigenvector* [5], an eigenvector corresponding to the second smallest eigenvalue of a Laplacian matrix. The Fiedler eigenvector is then used in [3] to derive a subgradient algorithm that increases the algebraic connectivity of a graph. However, the method in [9] is not scalable: In the task of estimating Fiedler eigenvector, each agent has to keep $O(n)$ estimator states, communicate with each neighbor $O(n^2)$ length of message and perform $O(n^2)$ computation at each state update. Moreover, the initialization step in [9] is not entirely decentralized.

In this paper we are concerned with controlling the global connectivity of the network (as in the second approach above) using only local communication and decentralized computations (as in the first approach above). The key component in our solution is a *decentralized power iteration* algorithm that enables each agent i to compute x^i, which is an estimate of the i-th component of the Fiedler eigenvector. This algorithm is entirely decentralized and scalable: Each agent keeps only $O(1)$ estimator states, and the communication and computational load at each state update is proportional to its local degree of connection. Each agent uses x^i to estimate the algebraic connectivity of a graph. Each agent also uses x^i in a decentralized controller that maintains the global connectivity of the graph over time.

The rest of the paper is organized as follows. We summarize the necessary graph theoretical background in Section II. In Section III, we first review the centralized discrete-time

This work was supported in part by NSF grants ECS-0601661 and IIS-0308224.

Peng Yang and Kevin Lynch are with the Department of Mechanical Engineering, Northwestern University, Evanston, IL 60208-3111, USA `p-yang@northwestern.edu`, `kmlynch@northwestern.edu`.

Randy Freeman is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208-3118, USA `freeman@ece.northwestern.edu`.

Geoffrey Gordon is with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA `ggordon@cs.cmu.edu`.

Siddhartha Srinivasa and Rahul Sukthankar are with Intel Research, Pittsburgh, PA 15213, USA `siddh@cs.cmu.edu`, `rahuls@cs.cmu.edu`.

[1]Given a graph's adjacency matrix A, its k-connectivity matrix is defined as $I + A + \cdots + A^k$ for $k \in \{1, \dots, n\}$.

power iteration algorithm and then describe our modified continuous-time version. We further characterize the gain conditions that guarantee the correct convergence of this continuous-time power iteration algorithm. In Section IV, we describe a decentralized version of this continuous-time power iteration procedure and use it to estimate the connectivity of a graph. This algorithm is scalable: the computational complexity of each agent is only proportional to its number of connections in the network. A controller to maintain connectivity is proposed in Section V. Future research directions are outlined in Section VI and the full stability analysis of the continuous-time power iteration algorithm is given in the appendix.

II. Preliminaries

Given n mobile agents, we assume they can exchange information on an undirected communication network. For agent i, we denote its set of communication neighbors as $\mathcal{N}^i$. We denote the overall communication graph as G and the edge set as $E = \{(i,j) | \, j \in \mathcal{N}^i\}$. The *adjacency* matrix $A \in \mathbb{R}^{n \times n}$ is defined as

$$A_{ij} = \begin{cases} A_{ji} > 0 & \text{if } j \in \mathcal{N}^i, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The degree of each node is $d_i = \sum_{j=1}^{n} A_{ij}$ or $d = A\mathbf{1}$ where $\mathbf{1}$ is a column vector of all ones. The *degree* matrix is defined as $D = \text{diag}(d)$, and the *weighted Laplacian* matrix of the graph is defined as $L = D - A$. The unweighted Laplacian matrix $\overline{L}$ can be treated as a special case where $A_{ij} = 1$. The spectral properties of $\overline{L}$ have been shown to be critical in many multiagent applications, such as formation control [4], [6], consensus seeking [16] and direction alignment [8].

For the weighted Laplacian L, because we restrict the weights A_{ij} to be positive, the spectral properties of L are similar to those of $\overline{L}$ [13]. Specifically, we know

1) $L\mathbf{1} = 0$.
2) Given $\{\{\lambda_i\} | \, i = 1, \ldots, n\}$ as the spectrum of L, all the eigenvalues are real and they satisfie $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, and $\lambda_2 > 0$ if and only if the graph is connected. As in the unweighted case, we call λ_2 the *algebraic connectivity* of the graph.

III. Centralized Power Iteration

We want to design an algorithm to estimate the graph connectivity measure λ_2. To do this, we first estimate the corresponding eigenvector v_2 ($Lv_2 = \lambda_2 v_2$), which is then used to determine λ_2.

Throughout the rest of the paper, we use superscripts to index the agents and components of a vector, and subscripts to index eigenvalues, eigenvectors, and their estimates. For example, a Laplacian L has n eigenvalues $\lambda_1, \ldots, \lambda_n$ and n eigenvectors $v_1, \ldots, v_n$. The components of an eigenvector are $v_i = (v_i^1, \ldots, v_i^n)^T$. In addition, if $x \in \mathbb{R}^n$ is the network's estimate of the eigenvector v_2, then $x^i \in \mathbb{R}$ is the ith component of the estimate x, stored by agent i. We also write $\lambda_2^i \in \mathbb{R}$ for agent i's estimate of λ_2.

A. Discrete-time Power Iteration

Given a square matrix Q and its eigenvalue spectrum satisfying $|\mu_1| < |\mu_2| \cdots < |\mu_n|$, *power iteration* is an established iterative method to compute the eigenvalue μ_n and its associated eigenvector v_n [21]. Now assume instead of μ_n, we are interested in its second-largest eigenvalue μ_{n-1}. If we already know μ_n and v_n, we can estimate μ_{n-1} by running the power iteration on the deflated matrix

$$\widetilde{Q} = Q - v_n v_n^T. \tag{2}$$

Specifically this power iteration procedure is carried out in three steps. For a random initial vector w,

1) Deflation on Q: $\widetilde{Q} = Q - v_n v_n^T$.
2) Direction update: $x = \widetilde{Q} w$.
3) Renormalization: $w = \frac{x}{\|x\|}$. Then go to step 2.

This power iteration method converges linearly in the ratio μ_{n-2}/μ_{n-1}. Once it converges, w is the eigenvector corresponding to the second largest eigenvalue μ_{n-1} of Q. In the case of repeated eigenvalues where $\mu_{n-1} = \ldots = \mu_{n-k+1} > \mu_{n-k}$, the iteration converges in the ratio μ_{n-k}/μ_{n-1}. If $\mu_{n-1} = \ldots = \mu_1$, then any unit vector w is a solution.

B. Continuous-time Power Iteration

Inspired by the power iteration algorithm, we define a variant to find the second-smallest eigenvalue λ_2. To do this, we make two primary modifications to the algorithm. First, we modify the algorithm to run in continuous-time. Second, instead of performing the direction update step using a deflated matrix $\tilde{Q}$, we use a deflated matrix of $-L$. The deflation causes $-\lambda_2$ to be the leading (least negative) eigenvalue, and the negative sign results in convergence to the eigenvector v_2 corresponding to the eigenvalue with the minimum magnitude (as opposed to that with the maximum magnitude in the previous section).

Let $x = (x^1 \ldots x^n)^T \in \mathbb{R}^n$ be the estimate of the eigenvector v_2. The continuous-time algorithm has three parts:

1) Deflation: $\dot{x} = -\text{Ave}(\{x^i\})\mathbf{1}$.
2) Direction update: $\dot{x} = -Lx$.
3) Renormalization: $\dot{x} = -(\text{Ave}(\{(x^i)^2\}) - 1)x$.

where the function $\text{Ave}(\{q^i\}) \triangleq (\sum_i q^i)/n$. Step 1 drives x to the null space of $\mathbf{1}$, i.e., the space spanned by the eigenvectors $\{v_2, \ldots, v_n\}$. For most initial conditions the direction update in step 2 drives x towards the eigenvector direction associated with the largest eigenvalue of $-L$, which is 0. But if the state x belongs to the null space of $\mathbf{1}$, the direction update step will keep x in the null space, and drive x towards the eigenvector direction associated with the largest eigenvalue of the null space, which is $-\lambda_2$. Step 3 drives x towards the unit sphere.

In order to achieve the three steps simultaneously, we combine the three pieces in a linearly weighted fashion:

$$\dot{x} = -k_1 \text{Ave}(\{x^i\})\mathbf{1} - k_2 Lx - k_3(\text{Ave}(\{(x^i)^2\}) - 1)x \tag{3}$$

This can be rewritten as

$$\dot{x} = -\frac{k_1}{n}\mathbf{1}\mathbf{1}^T x - k_2 L x - k_3\Big(\frac{x^T x}{n} - 1\Big)x. \quad (4)$$

The weighted Laplacian matrix L is real symmetric, so it has an eigenvalue decomposition $L = T^T L^* T$ with $L^* = \mathrm{diag}(0, \lambda_2, \dots, \lambda_n)$ and T being an orthonormal matrix. It is easier to analyze system (4) under a new set of coordinates $y = (y^1 \dots y^n)^T = Tx$ where both matrices L and $\mathbf{1}\mathbf{1}^T$ can be simultaneously diagonalized:

$$\dot{y} = -k_1 \mathrm{diag}(1, 0, \dots, 0)y - k_2 L^* y - k_3\Big(\frac{y^T y}{n} - 1\Big)y. \quad (5)$$

Denoting $\widetilde{L}^* = \mathrm{diag}\{k_1/k_2, \lambda_2, \dots, \lambda_n\}$, the system (5) can be rewritten as

$$\dot{y} = -k_2 \widetilde{L}^* y - k_3\Big(\frac{y^T y}{n} - 1\Big)y. \quad (6)$$

The following theorem shows that for suitable gain conditions on k_1, k_2 and k_3, system (3) is convergent from almost all initial conditions to an eigenvector $\tilde{v}_2$ corresponding to the eigenvalue $-\lambda_2$.

Theorem 1: Given any initial condition $x(t_0)$ and positive gains $k_1, k_2, k_3 > 0$, as long as $y^2(t_0) \neq 0$, the gain conditions

$$k_1 > k_2\lambda_2 \quad (7)$$

$$k_3 > k_2\lambda_2 \quad (8)$$

are necessary and sufficient for system (4) to converge to an eigenvector $\tilde{v}_2$ corresponding to the eigenvalue $-\lambda_2$ of the weighted Laplacian matrix $-L$ satisfying $\|\tilde{v}_2\| = \sqrt{n\Big(\frac{k_3 - k_2\lambda_2}{k_3}\Big)}$.

Proof: See the Appendix. ■

Next we modify the continuous-time power iteration (3) so that it can be decentralized over the graph. In the decentralized algorithm, no single agent maintains an estimate of the entire eigenvector $\tilde{v}_2$; instead, agent i maintains the single component x^i of the network's estimate x of $\tilde{v}_2$. This is sufficient to maintain an estimate λ_2^i of λ_2.

IV. Decentralized Power Iteration and Connectivity Estimation

To obtain a decentralized version of the power iteration algorithm, we first note that it is possible for each agent to satisfy the gain conditions (7) and (8) without knowing the graph topology. We know

$$\sum_i \lambda_i = \mathrm{trace}(L) = \sum A_{ij} \leq n(n-1)\max A_{ij}.$$

Additionally, in our edge weighting scheme introduced in Section V, we have $A_{ij} \leq 1$. Therefore each agent can satisfy (7) and (8) by choosing $k_3, k_1 > n(n-1)k_2$ (assuming n is known to every agent).

Next we point out that the matrix iteration $\dot{x} = -Lx$ is a naturally decentralized operation, and its implementation only requires local communication.

The last obstacle to decentralize the continuous-time power iteration (3) is the averaging operation $\mathrm{Ave}(\cdot)$. We can use the *PI average consensus estimator* [7] to decentralize this averaging operation. As there are two averaging functions in (3), we need two consensus estimators. Average consensus estimators allow n agents, each of which measures some time-varying scalar $\alpha^i(t)$, to compute an approximation of $\overline{\alpha}(t) = \frac{1}{n}\sum_i \alpha^i(t)$ using only local communication. The PI estimator has the form (see [7] for details):

$$\begin{aligned}\dot{z}^i =& \gamma(\alpha^i - z^i) - K_P \sum_{j\in\mathcal{N}^i}\big[z^i - z^j\big] \\ &+ K_I \sum_{j\in\mathcal{N}^i}\big[w^i - w^j\big]\end{aligned} \quad (9)$$

$$\dot{w}^i = -K_I \sum_{j\in\mathcal{N}^i}\big[z^i - z^j\big]. \quad (10)$$

Here z^i is the average estimate, $\gamma > 0$ is the rate new information replaces old information, $\mathcal{N}^i$ contains all one-hop neighbors of agent i in the communication network, and K_P, K_I are estimator gains. When the network is connected, the estimator error is $e^i(t) = y^i(t) - \frac{1}{n}\sum_{i=1}^n \alpha^i(t)$ for each agent i. Compared to other dynamic average consensus estimators [19], this PI consensus estimator has several advantages. First, it approaches a ball around zero whose size is related to the rate of change of the input, with constant input producing errors that decay exponentially to zero [7]. For the high-pass estimator [19], zero steady-state error requires extra bookkeeping to keep track which communication links are active. Besides, intermittent communication noise or drops cause the high-pass filters to drift, whereas a "forgetting" factor in the PI filter results in a stable filter from communication noise to errors relative to the solution manifold.

In the decentralized implementation of (3), agent i maintains a scalar x^i (which converges to the i-th component of the eigenvector $\tilde{v}_2$) and four consensus estimator states $\{z^{i,1}, w^{i,1}, z^{i,2}, w^{i,2}\}$ ($z^{i,1}$ and $z^{i,2}$ are agent i's estimates for $\mathrm{Ave}(\{x^i\})$ and $\mathrm{Ave}(\{(x^i)^2\})$ respectively) and receives from communication its neighbors' $\{x^j, z^{j,1}, w^{j,1}, z^{j,2}, w^{j,2}\}$ for all $j \in \mathcal{N}^i$. There are two ways to estimate the connectivity measure λ_2. First, noticing $-L\tilde{v}_2 = \lambda_2 v_2$, agent i can estimate λ_2 as

$$\lambda_2^i = -\frac{\sum_{j\in\mathcal{N}^i} L_{ij}x^j}{x^i} \quad (11)$$

when $x^i \neq 0$. This estimate is nonsmooth when x^i passes through zero, however. Therefore we use a second method, based on Theorem 1, which says that $z^{i,2} \rightarrow \frac{\|\tilde{v}_2\|^2}{n} = \frac{k_3 - k_2\lambda_2}{k_3}$. Agent i can therefore compute its estimate of λ_2 as

$$\lambda_2^i = \frac{k_3}{k_2}(1 - z^{i,2}). \quad (12)$$

Example 1: We simulated the eigenvalue estimation algorithm over the 5-node constant graph (Fig. 1), where the weights are set as $A_{ij} = 1$. The eigenvalue spectrum of its Laplacian matrix is $\{0, 0.83, 2.69, 4.00, 4.48\}$. The gains for the two PI average consensus estimators are $\gamma = 25, K_P = 50, K_I = 10$ and the gains for the eigenvector estimator are

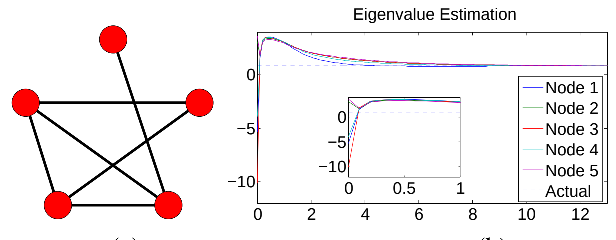

Fig. 1. (a) A five-node network with all weights equal to 1. Nodes are counter-clockwise numbered from 1 to 5 starting from the top node. (b) Eigenvalue estimation through equation (12). The initial eigenvalue estimate for each agent is randomized.

$k_1 = 6, k_2 = 1, k_3 = 20$, satisfying (7) and (8). Fig. 1 (b) shows the estimated λ_2^i for each node i.

V. Control to Maintain Connectivity

In this section we show how the connectivity estimator can be applied in a connectivity-control algorithm. We start by showing one additional property of λ_2.

Lemma 2: Given any positively weighted graph G, λ_2 is a nondecreasing function of each weight A_{ij}.

Remark 1: This lemma is easily demonstrated from the following equivalent definition of λ_2:

$$\lambda_2 = \min_{x \perp \mathbf{1}, x \neq 0} \frac{x^T L x}{x^T x} = \min_{x \perp \mathbf{1}, x \neq 0} \frac{\sum_{(i,j)\in E} A_{ij}(x^i - x^j)^2}{x^T x}. \tag{13}$$

Based on this property, we can choose a weight function A_{ij} that is position-dependent. Then we can design connectivity-maintaining motion controllers, moving the agents to increase the connectivity of the network.

If the maximal reliable inter-agent communication distance is r, one simple weighting choice is

$$A_{ij} = \begin{cases} e^{-\|p^i - p^j\|_2^2/2\sigma^2} & \text{if } \|p^i - p^j\|_2 \leq r, \\ 0 & \text{otherwise.} \end{cases}$$

The weight decreases as the inter-agent distance gets larger. We choose the parameter σ to satisfy a threshold condition $e^{-r^2/2\sigma^2} = \epsilon$, with ϵ being a small predefined threshold.

We know $\lambda_2 > 0$ for connected graphs, and based on Lemma 2, λ_2 increases as the graph adds more links or as individual link weights increase as two agents come closer. We can design a gradient controller where each node moves to maximize λ_2, and this will in effect maintain the connectivity of a graph. The gradient controller in [24] was designed based on a similar idea. In that paper, each node moves to maximize the determinant of the deflated Laplacian matrix of a graph, in effect guaranteeing the algebraic connectivity λ_2 is bounded away from 0.

Next we derive the analytical form of the gradient controller for fully-actuated first-order agents. We use the normalized eigenvector corresponding to λ_2 to make the gradient of λ_2 easier to derive. Given the normalized eigenvector $\hat{v}_2$ ($\|\hat{v}_2\| = 1$) corresponding to λ_2, the differential of λ_2 is

$$\begin{aligned} \mathbf{d}\lambda_2 &= \mathbf{d}(\hat{v}_2^T L \hat{v}_2) \\ &= \mathbf{d}\hat{v}_2^T L v_2 + \hat{v}_2^T \mathbf{d}L \hat{v}_2 + \hat{v}_2^T L \mathbf{d}\hat{v}_2. \end{aligned} \tag{14}$$

Because $L^T = L$, we know that

$$\hat{v}_2^T L \mathbf{d}\hat{v}_2 = \mathbf{d}\hat{v}_2^T L v_2 = \lambda_2 \mathbf{d}\hat{v}_2^T \hat{v}_2 = \frac{1}{2}\mathbf{d}(\hat{v}_2^T \hat{v}_2) = 0. \tag{15}$$

Based on (14) and (15), the gradient controller for agent k is

$$u^k = \dot{p}^k = \frac{\partial \lambda_2}{\partial p^k} = \hat{v}_2^T \frac{\partial L}{\partial p^k} \hat{v}_2. \tag{16}$$

Next we replace the $\hat{v}_2$ in (16) with the $\tilde{v}_2$ in Theorem 1, which scales the control effort but does not change its direction:

$$u^k = \tilde{v}_2^T \frac{\partial L}{\partial p^k} \tilde{v}_2 = \sum_{(i,j)\in E} \frac{\partial A_{ij}}{\partial p^k} (\tilde{v}_2^i - \tilde{v}_2^j)^2. \tag{17}$$

Since we have defined $A_{ij} = e^{-\|p^i - p^j\|_2^2/2\sigma^2}$, we can compute

$$\frac{\partial A_{ij}}{\partial p^i} = -A_{ij}(p^i - p^j)/\sigma^2 \qquad i \neq j \tag{18}$$
$$\frac{\partial A_{ij}}{\partial p^j} = A_{ij}(p^i - p^j)/\sigma^2 \qquad i \neq j \tag{19}$$
$$\frac{\partial A_{ii}}{\partial p^i} = 0 \tag{20}$$
$$\frac{\partial A_{ij}}{\partial p^k} = 0 \qquad k \neq i, j. \tag{21}$$

Plugging (18)-(21) into (17), we get

$$\begin{aligned} u^k &= \sum_{(k,j)\in E} \frac{\partial A_{kj}}{\partial p^k} (\tilde{v}_2^k - \tilde{v}_2^j)^2 \\ &= \sum_{(k,j)\in E} -A_{kj} (\tilde{v}_2^k - \tilde{v}_2^j)^2 \frac{p^k - p^j}{\sigma^2}. \end{aligned} \tag{22}$$

Compared to the eigenvector estimators (11) and (12), the implementation of (22) requires agent k to additionally obtain its neighbors' positions $\{p^j, j \in \mathcal{N}^i\}$. Agent k approximates the exact $\tilde{v}_2^k, \tilde{v}_2^j$ with the estimates x^k, x^j, yielding the final control law:

$$u^k = \sum_{(k,j)\in E} -A_{kj}(x^k - x^j)^2 \frac{p^k - p^j}{\sigma^2}. \tag{23}$$

Example 2: We simulated the connectivity-maintaining algorithm over a randomly-generated six-node network. The communication radius is $r = 20$ and we set the threshold $\epsilon = 0.01$. In this network, the three big nodes are leaders. They all follow the same sinusoidal motion model $\dot{p}_x^i(t) = -0.2$, $\dot{p}_y^i(t) = 0.5\cos(p_x^i)$ with different initial configurations. The three small nodes run (23) to move along with the leaders and maintain graph connectivity.

The gains for the two average consensus estimators are $\gamma = 100, K_P = 50, K_I = 200$ and the gains for the eigenvector estimator are $k_1 = 18, k_2 = 3, k_3 = 60$. We choose the consensus and eigenvector estimator gains to approximately achieve a time-scale separation: the time constant of consensus estimation is significantly less than

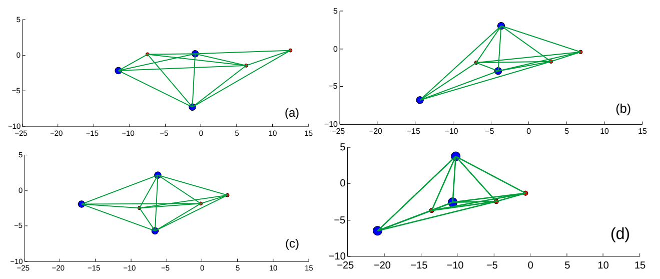

Fig. 2. Snapshots of the agents during motion: $(a)\, t = 0$; $(b)\, t = 14$; $(c)\, t = 27$; $(d)\, t = 47$.

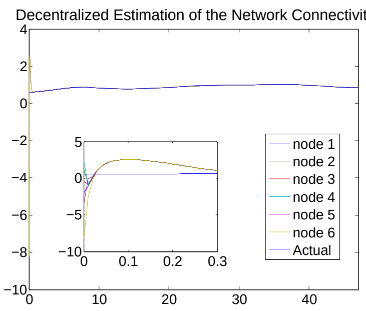

Fig. 3. Each agent's estimate of the the graph connectivity λ_2 over time. All agents' estimates converge to the true algebraic connectivity of the graph within a few seconds.

the time constant of eigenvector estimation, which is significantly less than the time constant of the motion controller. Fig. 2 shows four snapshots of these nodes during the motion and Fig. 3 shows the estimated λ_2^i of each node i during the motion. A video of the simulation is available at `http://lims.mech.northwestern.edu/projects/swarm/connect.wmv`.

VI. Future Work

In this paper we present a decentralized power iteration algorithm that agent i uses to estimate its component $\tilde{v}_2^i$ of the eigenvector $\tilde{v}_2$ of the Laplacian matrix L. We further show how each agent uses the estimate of $\tilde{v}_2^i$ to estimate the graph connectivity λ_2 and choose a motion direction to increase λ_2. One possible extension of the work is to modify the motion controller so that a rigorous small-gain stability condition can be derived for the system of coupled estimators and controllers as in [6].

Appendix

In the appendix we analyze the stability properties of system (3). We show the boundedness of all trajectories, characterize the equilibrium sets and their local stability, and finally give a proof of Theorem 1.

Proposition 3: Given any initial condition $x(t_0)$ and any positive gains $k_1, k_2, k_3 > 0$, the system trajectory remains bounded over time:

$$\|x(t)\| \leq \max\{\|x(t_0)\|, \sqrt{n}\}. \tag{24}$$

Proof: Defining $V_1 = x^T x = y^T y$, we have

$$\begin{aligned} \dot{V}_1 &= 2y^T \dot{y} \qquad (25) \\ &= 2y^T[-k_1 \mathrm{diag}(1, 0, \ldots, 0) - k_2 L^* - k_3\Big(\frac{y^T y}{n} - 1\Big)I]y. \end{aligned}$$

If $\|x(t_0)\| > \sqrt{n}$, then $k_3(\frac{y^T y}{n} - 1) > 0$ and $\dot{V}_1 < 0$ until $\|x(t)\| \leq \sqrt{n}$. If $\|x(t_0)\| \leq \sqrt{n}$, then $\|x(t)\| \leq \sqrt{n}$ for all $t > 0$. ∎

The following two propositions completely characterize the equilibrium sets of system (3) and their local stability properties.

Proposition 4: System (3) has an equilibrium point $x = 0$, and it is locally unstable when $k_3 > k_2\lambda_2$.

Proof: It is easy to verify that $x = 0$ (or $y = 0$) is an equilibrium state of system (3). Linearizing the equivalent system equation (6) around the point $y = \tilde{y}$ we get

$$\dot{y} = [-k_2 \tilde{L}^* - k_3\Big(\frac{\tilde{y}^T \tilde{y}}{n} - 1 + 2\frac{\tilde{y}\tilde{y}^T}{n}\Big)I]y. \tag{26}$$

Plugging in $\tilde{y} = 0$, equation (26) is simplied to $\dot{y} = [k_3 - k_2\tilde{L}^*]y$. The gain condition $k_3 > k_2\lambda_2$ makes the equilibrium point $y = 0$ locally unstable, at least in one direction. ∎

Now we proceed to investigate the non-zero equilibrium points of system (3).

Proposition 5: When the gain conditions (7), (8) are satisfied, system (3) has n (when $k_3 > k_1$) or $n - 1$ (when $k_3 \leq k_1$) pairs of distinct non-zero equilibrium points $\{y_i \mid 1 \leq i \leq n\}$ where

$$y_1 = \Big(\pm\sqrt{n(\frac{k_3 - k_1}{k_3}}), 0, \ldots, 0\Big)^T, \text{ if } k_3 > k_1; \tag{27}$$

and $\{y_i \mid 2 \leq i \leq n\}$ is

$$y_i^j = \begin{cases} 0 & \text{if } 2 \leq j \leq n, j \neq i, \\ \pm\sqrt{n(\frac{k_3 - k_2\lambda_i}{k_3})} & \text{if } j = i. \end{cases} \tag{28}$$

Additionally, among all the n or $n - 1$ pairs of equilibria, only y_2 is locally stable.

Proof: The insight here is that any nonzero equilibrium point y of system (6) has to be an eigenvector of the matrix $\tilde{L}^*$ with an associated eigenvalue $\frac{k_3}{k_2}(\frac{y^T y}{n} - 1)$. Furthermore, we know the n different unit eigenvectors for the diagonal matrix $\tilde{L}^* \in \mathbb{R}^{n\times n}$. Therefore, we can solve for all the eigenvectors of the system (6) that are also equilibria of the system. There are n such eigenvectors in total, described in (27) and (28). Additionally, we use the linearized models (26) to check the local stability of every y_i. For y_1, its eigenvalue spectrum $\{\mu_1^j \mid j = 1, \ldots, n\}$ is

$$\begin{cases} \mu_1^1 = -2(k_3 - k_1) & \text{if } j = 1, \\ \mu_1^j = k_1 - k_2\lambda_j & \text{if } j = 2, \ldots, n. \end{cases} \tag{29}$$

Since at least $\mu_1^2 > 0$, y_1 is locally unstable. Similarly for the equilibrium point y_i, $i = 2, \ldots, n$, its eigenvalue spectrum $\{\mu_i^j \mid j = 2, \ldots, n\}$ is

$$\begin{cases} \mu_i^1 = k_2\lambda_i - k_1 & \text{if } j = 1, \\ \mu_i^j = k_2(\lambda_i - \lambda_j) & \text{if } j = 2, \ldots, n, j \neq i, \\ \mu_i^i = -2(-k_2\lambda_i + k_3) & \text{if } j = i. \end{cases} \tag{30}$$

Because $0 < \lambda_2 \leq \cdots \leq \lambda_n$, y_i is unstable for any $i > 2$ (at least in some directions), and y_2 is stable. ■

Finally we give a proof for the near-global convergence result stated in Theorem 1.

It is useful to write out equation (6) in its scalar form:

$$\dot{y}^1 = (-k_1 - k_3\Big(\frac{y^T y}{n} - 1\Big))y^1 \quad (31)$$

$$\dot{y}^2 = (-k_2\lambda_2 - k_3\Big(\frac{y^T y}{n} - 1\Big))y^2 \quad (32)$$

$$\vdots$$

$$\dot{y}^n = (-k_2\lambda_n - k_3\Big(\frac{y^T y}{n} - 1\Big))y^n. \quad (33)$$

We first notice that the value of each component y^i will not change its sign over time and if $y^i(t_0) = 0$, $y^i(t)$ remains zero. Next we present the complete proof of the main theorem.

Proof: (Sufficiency) Let us first consider y^1. If $y^1(t_0) = 0$, then $y^1(t) = 0$ for all t. If $y^1(t_0) \neq 0$, combining (31) and (32) we get

$$\frac{d}{dt}(\ln\frac{y^2}{y^1}) = \frac{d}{dt}(\ln y^2) - \frac{d}{dt}(\ln y^1) = k_1 - k_2\lambda_2 > 0 \quad (34)$$

which implies $y^2/y^1 \to \infty$. We know y^2 is bounded from Theorem 3, therefore $y^1 \to 0$. The cases are similar for $y^i, i > 2$. If $y^i(t_0) = 0$, then $y^i(t) = 0$ for all t. If $y^i(t_0) \neq 0$, then $y^2/y^i \to \infty$ and $y^i \to 0$.

Therefore over time equation (32) is reduced to $\dot{y}^2 = (-k_2\lambda_2 - k_3(\frac{(y^2)^2}{n} - 1))y^2$. When (8) holds, this scalar dynamical system can be rewritten as

$$\dot{y}^2 = \frac{k_3}{n}(\sqrt{n\Big(\frac{k_3 - k_2\lambda_2}{k_3}\Big)} + y^2)(\sqrt{n\Big(\frac{k_3 - k_2\lambda_2}{k_3}\Big)} - y^2)y^2. \quad (35)$$

We see that $y^2 \to \pm\sqrt{n\Big(\frac{k_3 - k_2\lambda_2}{k_3}\Big)}$ depending on the initial condition $y^2(t_0)$ and the equilibrium point $y^2 = 0$ is unstable.

(Necessity) When $y^2 \to \pm\sqrt{n\Big(\frac{k_3 - k_2\lambda_2}{k_3}\Big)}$ obviously condition (8) holds. Now we suppose the condition $k_1 \leq k_2\lambda_2$ holds. If $k_1 < k_2\lambda_2$, using the same argument method in (34), $y^1/y^2 \to \infty$ and therefore $y^2 \to 0$, which is a contradiction. If $k_1 = k_2\lambda_2$, then $\frac{d}{dt}(\ln\frac{y^1}{y^2}) = 0$ and y^1/y^2 is a constant c. For initial conditions $y^1(t_0) \neq 0$, $c = y^1(t_0)/y^2(t_0) \neq 0$, therefore y cannot converge to y_2 where $y_2^1/y_2^2 = 0$, which is also a contradiction. Therefore, the gain condition (7) must hold. ■

Remark 2: In case of repeated eigenvalues $\lambda_2 = \cdots = \lambda_k < \lambda_{k+1}$, Theorem 1 still holds. In this case all trajectories with $y^2(t_0) \neq 0$ converge to an equilibrium point on the k-dimensional manifold $\{y| \|y\| = \sqrt{n\Big(\frac{k_3 - k_2\lambda_2}{k_3}\Big)}, y^1 = 0, y^i = 0, \forall i > k\}$.

References

[1] C. Belta and V. Kumar. Abstraction and control for groups of robots. *IEEE Transactions on Robotics*, 20(5):865–875, Oct. 2004.

[2] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[3] M. De Gennaro and A. Jadbabaie. Decentralized control of connectivity for multi-agent systems. *IEEE International Conference on Decision and Control*, pages 3628–3633, 2006.

[4] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, Sep 2004.

[5] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.

[6] R. A. Freeman, P. Yang, and K. M. Lynch. Distributed estimation and control of swarm formation statistics. In *American Control Conference*, 2006.

[7] R. A. Freeman, P. Yang, and K. M. Lynch. Stability and convergence properties of dynamic consensus estimators. In *IEEE International Conference on Decision and Control*, 2006.

[8] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, Jun 2003.

[9] D. Kempe and F. McSherry. A decentralized algorithm for spectral analysis. *Journal of Computer and System Sciences*, 74(1):70–83, 2008.

[10] N. E. Leonard, D. Paley, F. Lekien, R. Sepulchre, D. Fratantoni, and R. Davis. Collective motion, sensor networks, and ocean sampling. In *Proceedings of the IEEE Special Issue on Networked Control Systems*, volume 95, pages 48–74, Jan 2007.

[11] K. Lynch, I. Schwartz, P. Yang, and R. Freeman. Decentralized environmental modeling by mobile sensor networks. *IEEE Transactions on Robotics*, 24(3):710–724, 2008.

[12] S. Martínez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42:661–668, Apr 2006.

[13] B. Mohar. The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, 2:871–898, 1991.

[14] G. Notarstefano, K. Savla, F. Bullo, and A. Jadbabaie. Maintaining limited-range connectivity among second-order agents. In *American Control Conference*, pages 2124–2129, 2006.

[15] S. Oh and S. Sastry. Tracking on a graph. In *Proc. of the Fourth International Conference on Information Processing in Sensor Networks (IPSN05)*, Los Angeles, CA, April 2005.

[16] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Automat. Contr.*, 49(9):1520–1533, Sept. 2004.

[17] S. Simic and S. Sastry. Distributed environmental monitoring using random sensor networks. In *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks*, pages 582–592, Palo Alto, California, 2003.

[18] D. P. Spanos and R. M. Murray. Robust connectivity of networked vehicles. In *IEEE International Conference on Decision and Control*, 2004.

[19] D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Dynamic consensus on mobile networks. In *IFAC World Congress*, 2005.

[20] S. Susca, S. Martínez, and F. Bullo. Monitoring environmental boundaries with a robotic sensor network. In *American Control Conference*, pages 2072–2077, 2006.

[21] L. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[22] P. Yang, R. Freeman, and K. Lynch. Distributed cooperative active sensing using consensus filters. *Proc. of 2007 IEEE International Conference on Robotics and Automation, Rome, Italy*, pages 405–410, 2007.

[23] M. M. Zavlanos and G. J. Pappas. Controlling connectivity of dynamic graphs. In *IEEE Conference on Decision and Control*, pages 6388–6393, Dec 2005.

[24] M. M. Zavlanos and G. J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 23(4):812–816, Aug 2007.

[25] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.