Automatica 46 (2010) 390–396

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

Decentralized estimation and control of graph connectivity for mobile sensor networks☆

P. Yang [a], R.A. Freeman [b,*], G.J. Gordon [c], K.M. Lynch [a], S.S. Srinivasa [d], R. Sukthankar [d]

[a] Department of Mechanical Engineering, Northwestern University, Evanston, IL 60208-3111, United States

[b] Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208-3118, United States

[c] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, United States

[d] Intel Research, Pittsburgh, PA 15213, United States

ARTICLE INFO

Article history:
Received 17 September 2008
Received in revised form
20 July 2009
Accepted 5 November 2009
Available online 11 December 2009

Keywords:
Connectivity
Decentralized control
Graph theory
Numerical algorithms
Autonomous mobile robots

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.

1. Introduction

A *mobile sensor network* consists of n mobile sensors (or agents), some of which are connected by communication links along which information flows. Applications for mobile sensor networks include target tracking (Martínez & Bullo, 2006; Oh & Sastry, 2005; Yang, Freeman, & Lynch, 2007; Zhao, Shin, & Reich, 2002), formation and coverage control (Belta & Kumar, 2004; Cortés, Martínez, Karatas, & Bullo, 2004; Fax & Murray, 2004; Freeman, Yang, & Lynch, 2006a), environmental monitoring (Leonard et al., 2007; Lynch, Schwartz, Yang, & Freeman, 2008; Simic & Sastry, 2003; Susca, Martínez, & Bullo, 2006), and several others. These applications exploit the sensors' ability to dynamically reposition themselves to maximize the collective information gained by the network. For these cooperative sensing applications, it is often also 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 schemes, and control of global connectivity measures using centralized schemes.

The first approach focuses on devising decentralized controllers that enable each agent to maintain local connectivity. For discrete-time second-order agents, a feasible control space is computed in Notarstefano, Savla, Bullo, and Jadbabaie (2006) for each agent to maintain all existing pairwise connections. In comparison, in Spanos and Murray (2004) 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 the strict maintenance of local connectivity may be overly restrictive.

The second approach in De Gennaro and Jadbabaie (2006) and Zavlanos and Pappas (2005, 2007) uses global connectivity measures such as *algebraic connectivity* (Fiedler, 1973). Given a graph, a *k-connectivity matrix*[1] is computed in Zavlanos and Pappas (2005).

☆ This work was supported in part by NSF grants ECS-0601661 and IIS-0308224, and by the Office of Naval Research. The material in this paper was partially presented at the 2008 American Control Conference at Seattle, Washington, USA, June 11–13, 2008. This paper was recommended for publication in revised form by Associate Editor Dragan Nešić under the direction of Editor Andrew R. Teel.

* Corresponding author. Tel.: +1 847 467 2606; fax: +1 847 491 4455.
E-mail addresses: p-yang@northwestern.edu (P. Yang), freeman@eecs.northwestern.edu (R.A. Freeman), ggordon@cs.cmu.edu (G.J. Gordon), kmlynch@northwestern.edu (K.M. Lynch), siddh@cs.cmu.edu (S.S. Srinivasa), rahuls@cs.cmu.edu (R. Sukthankar).

0005-1098/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2009.11.012

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

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 Zavlanos and Pappas (2007) uses the fact that connectedness of the graph is equivalent to the determinant of the *reduced Laplacian matrix* being positive. However, computing the k-connectivity matrix and the determinant of a reduced Laplacian matrix are both centralized procedures. A method to compute the k leading eigenvectors of an $n \times n$ matrix is proposed in Kempe and McSherry (2008). In De Gennaro and Jadbabaie (2006) this method is used to compute the *Fiedler eigenvector* (Fiedler, 1973), an eigenvector corresponding to the second-smallest eigenvalue of a Laplacian matrix. The Fiedler eigenvector is then used in De Gennaro and Jadbabaie (2006) to derive a subgradient algorithm that increases the algebraic connectivity of a graph. However, the method in Kempe and McSherry (2008) is not scalable: In the task of estimating the Fiedler eigenvector, each agent has to keep $O(n)$ estimator states, communicate with each neighbor messages of length $O(n^2)$ and perform $O(n^2)$ computations at each state update. Moreover, the initialization step in Kempe and McSherry (2008) is not decentralized.

In this paper we focus on 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 ith component of the Fiedler eigenvector. This algorithm is entirely decentralized and scalable: each agent updates 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 the network. 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 2. In Section 3, we first review the centralized discrete-time 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 4, 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 5. Future research directions are outlined in Section 6 and the full stability analysis of the continuous-time power iteration algorithm is given in the Appendix.

2. 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 $\bar{L}$ can be treated as a special case where $A_{ij} = 1$ for $j \in \mathcal{N}^i$. The spectral properties of $\bar{L}$ have been shown to be critical in many multi-agent applications, such as formation control (Fax & Murray, 2004; Freeman et al., 2006a), consensus seeking (Olfati-Saber & Murray, 2004) and direction alignment (Jadbabaie, Lin, & Morse, 2003).

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 $\bar{L}$ (Mohar, 1991). 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 satisfy $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, 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.

3. 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.

3.1. Discrete-time power iteration

Given a square matrix Q with 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 (Trefethen & Bau, 1997). 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

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

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

(1) Deflation on Q: $\widetilde{Q} = Q - \mu_n v_n v_n^T$.
(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 $\mu_{n-1} = \cdots = \mu_{n-k+1} > \mu_{n-k}$, the iteration converges in the ratio μ_{n-k}/μ_{n-1}. If $\mu_{n-1} = \cdots = \mu_1$, then any unit vector w is a solution.

3.2. Continuous-time power iteration

Inspired by the power iteration algorithm, we define a continuous-time variant, and use it to find the second-smallest eigenvalue λ_2 (and corresponding eigenvector v_2) of the weighted Laplacian L. Modifications necessary for continuous time are described below. Because all eigenvalues of L are nonnegative, to find the second-smallest eigenvalue, we can run power iteration on $I - \alpha L$ for some sufficiently small $\alpha > 0$. Since $L\mathbf{1} = 0$, we know that $\mathbf{1}$ is the eigenvector corresponding to the largest eigenvalue of

$I - \alpha L$. So, deflation is simple: we just ensure that our estimated eigenvector has zero mean.

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

(1) Deflation: $\dot{x} = -\text{Ave}(\{x^i\})\mathbf{1}$.
(2) Direction update: $\dot{x} = -\alpha L x$.
(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 $\{v_2, \dots, v_n\}$. For most initial conditions the direction update in step 2 drives x towards the eigenvector $v_1 = \mathbf{1}$. But if the state x satisfies $\mathbf{1}^T x = 0$, then the direction update step will drive x towards the eigenvector v_2. Step 3 drives x towards a sphere of radius $\sqrt{n}$.

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

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

where k_1, k_2, k_3 are scalar control gains (and we have absorbed α into k_2). This equation can be rewritten as

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

The weighted Laplacian matrix L is real symmetric, so it has an eigenvalue decomposition $L = T^T L^* T$ with $L^* = \text{diag}(0, \lambda_2, \dots, \lambda_n)$ and T 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:

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

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

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

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 λ_2.

Theorem 1. *Given any initial condition $x(t_0)$ and positive gains $k_1, k_2, k_3 > 0$, as long as $v_2^T x(t_0) \neq 0$, the gain conditions*

$$k_1 > k_2 \lambda_2 \tag{7}$$

$$k_3 > k_2 \lambda_2 \tag{8}$$

are necessary and sufficient for system (4) to converge to an eigenvector $\tilde{v}_2$ corresponding to the eigenvalue λ_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. □

Remark 1. In case of repeated eigenvalues $\lambda_2 = \dots = \lambda_k < \lambda_{k+1}$, Theorem 1 still holds. In this case all trajectories with $v_2^T x(t_0) \neq 0$ converge to an equilibrium point on the k-dimensional manifold $\left\{y \,|\, \|y\| = \sqrt{n\Big(\frac{k_3 - k_2\lambda_2}{k_3}\Big)}, y^1 = 0, y^i = 0, \forall i > k\right\}$.

Next we modify the continuous-time power iteration (3) so that it is 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.

4. 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

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

Additionally, in our edge weighting scheme introduced in Section 5, we have $A_{ij} \leq 1$. Therefore each agent satisfies (7) and (8) by employing identical gains $k_3, k_1 > n(n-1)k_2$. If n may change with time, we can replace it by an upper-bound on n.

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 decentralizing the continuous-time power iteration (3) is the averaging operation $\text{Ave}(\cdot)$. We can use the *PI average consensus estimator* (Freeman, Yang, & Lynch, 2006b) 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 Freeman et al., 2006b, for details):

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

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

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. The estimator error is $\text{e}^i(t) = y^i(t) - \frac{1}{n}\sum_{i=1}^n \alpha^i(t)$ for each agent i. Compared to other dynamic average consensus estimators (e.g. Spanos, Olfati-Saber, & Murray, 2005), this consensus estimator has several advantages. First, when the network is connected, the error 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 (Freeman et al., 2006b). For the high-pass estimator (Spanos et al., 2005), zero steady-state error requires extra bookkeeping to keep track of which communication links are active. In addition, 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 runs two consensus estimators of the form (9)–(10), one with input $\alpha^{i,1} = x_i$ and states $(z^{i,1}, w^{i,1})$, and another with input $\alpha^{i,2} = (x^i)^2$ and states $(z^{i,2}, w^{i,2})$. Here $z^{i,1}$ is the agent's estimate of $\text{Ave}(\{x^i\})$ and $z^{i,2}$ is its estimate of $\text{Ave}(\{(x^i)^2\})$. Each agent i also runs the update law (3) but with $\text{Ave}(\{x^i\})$ and $\text{Ave}(\{(x^i)^2\})$ replaced by $z^{i,1}$ and $z^{i,2}$, respectively:

$$\dot{x}^i = -k_1 z^{i,1} - k_2 \sum_{j \in \mathcal{N}^i} A_{ij}(x^i - x^j) - k_3(z^{i,2} - 1)x^i. \tag{11}$$

In this implementation, agent i receives its neighbors' five variables $\{x^j, z^{j,1}, w^{j,1}, z^{j,2}, w^{j,2}\}$ for all $j \in \mathcal{N}^i$. If the consensus estimators are running fast enough relative to the update law (3), that is, if the gains γ, K_P, and K_I are large enough relative to k_1, k_2, and k_3, then we expect the resulting dynamics to converge semiglobally. Indeed, we see from (11) that peaking in the variables $z^{i,1}$ and $z^{i,2}$ will not cause finite escape times as might be possible in general

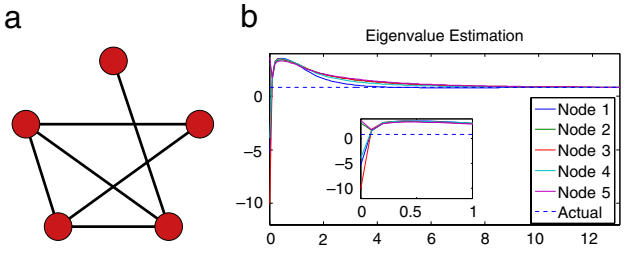


Fig. 1. (a) A five-node network with all link weights equal to 1. Nodes are numbered counter-clockwise from 1 to 5 starting from the top node. (b) Eigenvalue estimation through Eq. (13). The initial eigenvalue estimate for each agent is randomized. The inset plot shows the transient dynamics of the eigenvalue estimator.

nonlinear systems (Sussmann & Kokotović, 1991), so after a transient period the decentralized update law (11) will agree with the centralized one (3).

There are two ways to estimate λ_2. First, noticing $-L\tilde{v}_2 = \lambda_2 v_2$, agent i can estimate λ_2 as

$$\lambda_2^i = -\frac{\sum_{j\in\mathcal{N}^i} L_{ij}x^j}{x^i} \tag{12}$$

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

$$\lambda_2^i = \frac{k_3}{k_2}(1 - z^{i,2}). \tag{13}$$

Example 1. We simulated the eigenvalue estimation algorithm over the 5-node constant graph (Fig. 1), where the weights are set as $A_{ij} = 1$ for $j \in \mathcal{N}^i$. 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 $k_1 = 6$, $k_2 = 1$, $k_3 = 20$, satisfying (7) and (8). Fig. 1(b) shows the estimated λ_2^i for each node i as they converge to the correct eigenvalue of 0.83.

5. Control to maintain connectivity

In this section we show how the connectivity estimator can be applied in a connectivity-maintenance algorithm for fully-actuated point robots, where each robot's configuration is given by $p^i \in \mathbb{R}^d$. 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 2. This lemma is easily demonstrated from the following equivalent definition of λ_2:

$$\lambda_2 = \min_{x\perp\mathbf{1}, x\neq 0} \frac{x^T L x}{x^T x} = \min_{x\perp\mathbf{1}, x\neq 0} \frac{\sum_{(i,j)\in E} A_{ij}(x^i - x^j)^2}{x^T x}. \tag{14}$$

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.

Given a scalar r as the maximal reliable inter-agent communication distance, one simple weighting choice is

$$A_{ij} = \begin{cases} e^{-\|p^i - p^j\|_2^2/2\sigma^2} & \text{if } \|p^i - p^j\|_2 \leq r, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

The weight decreases as the inter-agent distance gets larger. We choose the scalar 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 Zavlanos and Pappas (2007) was designed based on a similar idea. In that paper, each node moves to maximize the determinant of the reduced 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

$$\begin{aligned} \mathbf{d}\lambda_2 &= \mathbf{d}(\hat{v}_2^T L \hat{v}_2) \\ &= \mathbf{d}\hat{v}_2^T L v_2 + \hat{v}_2^T \mathbf{d}L \hat{v}_2 + \hat{v}_2^T L \mathbf{d}\hat{v}_2. \end{aligned} \tag{16}$$

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

$$\hat{v}_2^T L \mathbf{d}\hat{v}_2 = \mathbf{d}\hat{v}_2^T L v_2 = \lambda_2 \mathbf{d}\hat{v}_2^T \hat{v}_2 = \frac{1}{2}\mathbf{d}(\hat{v}_2^T \hat{v}_2) = 0. \tag{17}$$

Based on (16) and (17), the gradient controller for agent k is

$$u^k = \dot{p}^k = \frac{\partial \lambda_2}{\partial p^k} = \hat{v}_2^T \frac{\partial L}{\partial p^k} \hat{v}_2. \tag{18}$$

Next we replace the $\hat{v}_2$ in (18) with the $\tilde{v}_2$ in Theorem 1, which scales the control effort but does not change its direction:

$$u^k = \tilde{v}_2^T \frac{\partial L}{\partial p^k} \tilde{v}_2 = \sum_{(i,j)\in E} \frac{\partial A_{ij}}{\partial p^k}(\tilde{v}_2^i - \tilde{v}_2^j)^2. \tag{19}$$

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

$$\frac{\partial A_{ij}}{\partial p^i} = -A_{ij}(p^i - p^j)/\sigma^2 \quad i \neq j \tag{20}$$

$$\frac{\partial A_{ij}}{\partial p^j} = A_{ij}(p^i - p^j)/\sigma^2 \quad i \neq j \tag{21}$$

$$\frac{\partial A_{ii}}{\partial p^i} = 0 \tag{22}$$

$$\frac{\partial A_{ij}}{\partial p^k} = 0 \quad k \neq i, j. \tag{23}$$

Plugging (20)–(23) into (19), we get

$$\begin{aligned} u^k &= \sum_{(k,j)\in E} \frac{\partial A_{kj}}{\partial p^k}(\tilde{v}_2^k - \tilde{v}_2^j)^2 \\ &= \sum_{(k,j)\in E} -A_{kj}(\tilde{v}_2^k - \tilde{v}_2^j)^2 \frac{p^k - p^j}{\sigma^2}. \end{aligned} \tag{24}$$

Implementation of (24) requires agent k to obtain its neighbors' positions $\{p^j, j \in \mathcal{N}^k\}$. 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:

$$\dot{p}^k = u^k = \sum_{(k,j)\in E} -A_{kj}(x^k - x^j)^2 \frac{p^k - p^j}{\sigma^2}, \tag{25}$$

with x^k and x^j coming from the estimators of Section 4.

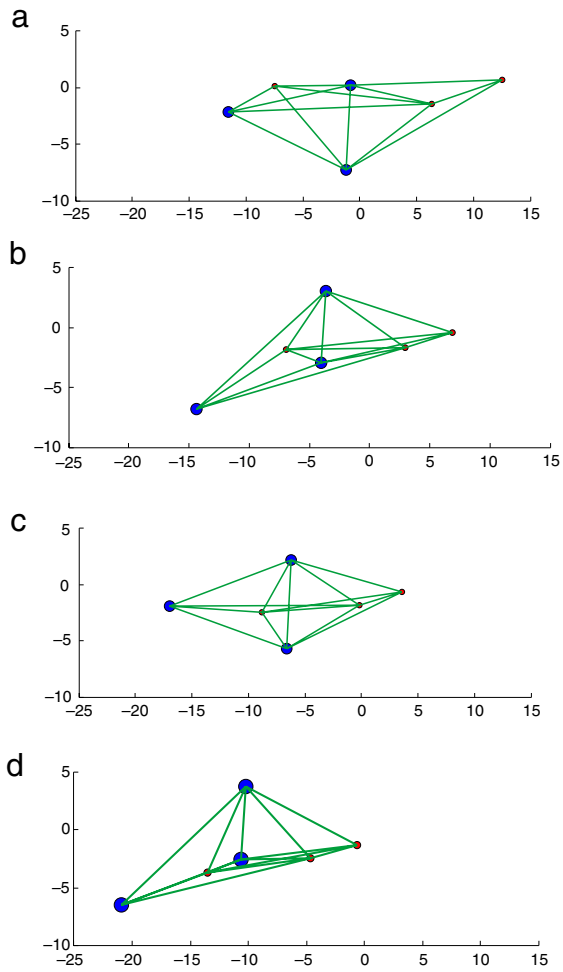


Fig. 2. Snapshots of the agents during motion: (a) $t = 0$; (b) $t = 14$; (c) $t = 27$; (d) $t = 47$.

Example 2. We simulated the connectivity-maintaining algorithm over a randomly-generated six-node network moving in a plane. The communication radius is $r = 20$ and we set the threshold $\epsilon = 0.01$. In this network, three nodes are leaders and three nodes are followers. The leaders, shown as larger dots in Fig. 2, use the same sinusoidal motion model $\dot{p}_x^i(t) = -0.2$, $\dot{p}_y^i(t) = 0.5\cos(p_x^i)$ from different initial configurations. The three follower agents run the control law (25) 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 $k_1 = 18, k_2 = 3, k_3 = 60$. For reasons given in Section 4, we choose the time constant of the consensus estimation to be significantly less than the time constant of the eigenvector estimation. For entirely analogous reasons, we choose this latter time constant to be significantly less than the time constant of the motion controller, so that after a transient period, the decentralized controller (25) will behave like the gradient controller (19). Again, it is clear from the linear dependence of the control law (25) on the positions p^k and p^j that peaking in the transient will not cause instabilities.

Typically, connectivity-maintenance is not the only objective of a mobile sensor network; the agents will also move to collect information, follow leader agents, etc. In general, the connectivity-maintenance controller (25) should be used in conjunction with some other control objective to prevent the agents from simply collapsing on each other. In the example above, the independent motion of the leader robots prevents this collapse. In practice, the controller (25) could be augmented by a collision-avoidance term, for example by modifying the link weights in (15) to be maximized at a nonzero distance between agents.

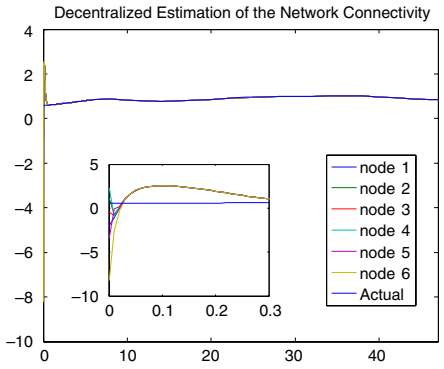


Fig. 3. Each agent's estimate of the graph connectivity λ_2 over time. All agents' estimates converge to the true algebraic connectivity of the graph within a few seconds.

6. 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 (see Fig. 3). In future work, we will explore the properties of discrete-time implementations of the estimator and controller, particularly considering packet drops, asynchronous updates, and time delays.

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. For simplicity, we assume throughout that λ_2 is an isolated eigenvalue of L (otherwise the proof needs some minor modifications).

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:*

$$\|x(t)\| \leq \max\{\|x(t_0)\|, \sqrt{n}\}. \tag{A.1}$$

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

$$\begin{aligned}\dot{V}_1 &= 2y^T\dot{y} \\ &= 2y^T\left[-k_1\mathrm{diag}(1,0,\ldots,0) - k_2L^* - k_3\left(\frac{y^Ty}{n} - 1\right)I\right]y.\end{aligned} \tag{A.2}$$

If $\|x(t_0)\| > \sqrt{n}$, then $k_3(\frac{y^Ty}{n} - 1) > 0$ 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 (6) around the point $y = \tilde{y}$ we get

$$\dot{y} = \left[-k_2\widetilde{L}^* - k_3\Big(\frac{\tilde{y}^T\tilde{y}}{n} - 1 + 2\frac{\tilde{y}\tilde{y}^T}{n}\Big)I\right]y. \tag{A.3}$$

Plugging in $\tilde{y} = 0$, Eq. (A.3) simplifies 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 nonzero 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 $k_3 \le k_1$) pairs of distinct nonzero equilibrium points $\{y_i \mid 1 \le i \le n\}$ where*

$$y_1 = \left(\pm\sqrt{n\left(\frac{k_3 - k_1}{k_3}\right)}, 0, \ldots, 0\right)^T, \quad \text{if } k_3 > k_1; \tag{A.4}$$

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

$$y_i^j = \begin{cases} 0 & \text{if } 2 \le j \le n, j \ne i, \\ \pm\sqrt{n\left(\dfrac{k_3 - k_2\lambda_i}{k_3}\right)} & \text{if } j = i. \end{cases} \tag{A.5}$$

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 of (6) has to be a real eigenvector of the matrix $\widetilde{L}^*$: setting $\dot{y} = 0$, we get $\widetilde{L}^*y = -\frac{k_3}{k_2}(\frac{y^Ty}{n} - 1)y$. In particular, if $v \ne 0$ satisfies $\widetilde{L}^*v = \lambda v$, then $y = \pm\sqrt{n\frac{k_3 - k_2\lambda}{k_3}}v$ is a nonzero equilibrium point iff the quantity inside the square root is positive. Furthermore, we know the n different unit eigenvectors for the diagonal matrix $\widetilde{L}^* \in \mathbb{R}^{n\times n}$. Therefore, we can solve for all of the equilibria of the system (6) by looking at the eigenvectors of $\widetilde{L}^*$. There are n such eigenvectors in total, described in (A.4) and (A.5). Additionally, we use the linearized models (A.3) to check the local stability of every y_i. For y_1, its eigenvalue spectrum $\{\mu_1^j \mid j = 1, \ldots, n\}$ is

$$\begin{cases} \mu_1^1 = -2(k_3 - k_1) & \text{if } j = 1, \\ \mu_1^j = k_1 - k_2\lambda_j & \text{if } j = 2, \ldots, n. \end{cases} \tag{A.6}$$

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

$$\begin{cases} \mu_i^1 = k_2\lambda_i - k_1 & \text{if } j = 1, \\ \mu_i^j = k_2(\lambda_i - \lambda_j) & \text{if } j = 2, \ldots, n,\ j \ne i, \\ \mu_i^i = -2(-k_2\lambda_i + k_3) & \text{if } j = i. \end{cases} \tag{A.7}$$

Because $0 < \lambda_2 \le \cdots \le \lambda_n$, 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 Eq. (6) in its scalar form:

$$\dot{y}^1 = \left(-k_1 - k_3\Big(\frac{y^Ty}{n} - 1\Big)\right)y^1 \tag{A.8}$$

$$\dot{y}^2 = \left(-k_2\lambda_2 - k_3\Big(\frac{y^Ty}{n} - 1\Big)\right)y^2 \tag{A.9}$$

$$\vdots$$

$$\dot{y}^n = \left(-k_2\lambda_n - k_3\Big(\frac{y^Ty}{n} - 1\Big)\right)y^n. \tag{A.10}$$

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 $y^1(t_0) \ne 0$, combining (A.8) and (A.9) we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\ln\frac{y^2}{y^1}\right) = \frac{\mathrm{d}}{\mathrm{d}t}(\ln y^2) - \frac{\mathrm{d}}{\mathrm{d}t}(\ln y^1) = k_1 - k_2\lambda_2 > 0 \tag{A.11}$$

which implies $y^2/y^1 \to \infty$. We know y^2 is bounded from Proposition 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 $y^i(t_0) \ne 0$, then $y^2/y^i \to \infty$ and $y^i \to 0$. Therefore over time Eq. (A.9) 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

$$\dot{y}^2 = \frac{k_3}{n}\left(\sqrt{n\Big(\frac{k_3 - k_2\lambda_2}{k_3}\Big)} + y^2\right) \times \left(\sqrt{n\Big(\frac{k_3 - k_2\lambda_2}{k_3}\Big)} - y^2\right)y^2. \tag{A.12}$$

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

(*Necessity*) When $y^2 \to \pm\sqrt{n\left(\frac{k_3 - k_2\lambda_2}{k_3}\right)}$ obviously condition (8) holds. Now we suppose the condition $k_1 \le k_2\lambda_2$ holds. If $k_1 < k_2\lambda_2$, using the same argument method in (A.11), $y^1/y^2 \to \infty$ and therefore $y^2 \to 0$, which is a contradiction. If $k_1 = k_2\lambda_2$, then $\frac{\mathrm{d}}{\mathrm{d}t}(\ln\frac{y^1}{y^2}) = 0$ and y^1/y^2 is a constant c. For initial conditions $y^1(t_0) \ne 0$, $c = y^1(t_0)/y^2(t_0) \ne 0$; 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. □

References

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

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

De Gennaro, M., & Jadbabaie, A. (2006). Decentralized control of connectivity for multi-agent systems. In *IEEE international conference on decision and control* (pp. 3628–3633).

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

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

Freeman, R. A., Yang, P., & Lynch, K. M. (2006a). Distributed estimation and control of swarm formation statistics. In *American control conference*.

Freeman, R. A., Yang, P., & Lynch, K. M. (2006b). Stability and convergence properties of dynamic consensus estimators. In *IEEE international conference on decision and control*.

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

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

Leonard, N. E., Paley, D., Lekien, F., Sepulchre, R., Fratantoni, D., & Davis, R. (2007). Collective motion, sensor networks, and ocean sampling. In *Proceedings of the IEEE special issue on networked control systems, Vol. 95* (pp. 48–74)..

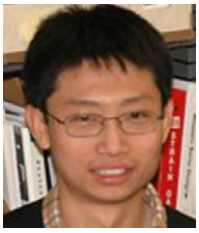
Lynch, K., Schwartz, I., Yang, P., & Freeman, R. (2008). Decentralized environmental modeling by mobile sensor networks. *IEEE Transactions on Robotics*, *24*(3), 710–724.

Martínez, S., & Bullo, F. (2006). Optimal sensor placement and motion coordination for target tracking. *Automatica*, *42*, 661–668.

Mohar, B. (1991). The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, *2*, 871–898.

Notarstefano, G., Savla, K., Bullo, F., & Jadbabaie, A. (2006). Maintaining limited-range connectivity among second-order agents. In *American control conference* (pp. 2124–2129).

Oh, S., & Sastry, S. (2005). Tracking on a graph. In *Proc. of the fourth international conference on information processing in sensor networks, IPSN05*. Los Angeles, CA.
Olfati-Saber, R., & Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, *49*(9), 1520–1533.
Simic, S., & Sastry, S. (2003). Distributed environmental monitoring using random sensor networks. In *Proceedings of the 2nd international workshop on information processing in sensor networks*, Palo Alto, CA (pp. 582–592).
Spanos, D. P., & Murray, R. M. (2004). Robust connectivity of networked vehicles. In *IEEE international conference on decision and control*.
Spanos, D. P., Olfati-Saber, R., & Murray, R. M. (2005). Dynamic consensus on mobile networks. In *IFAC world congress*.
Susca, S., Martínez, S., & Bullo, F. (2006). Monitoring environmental boundaries with a robotic sensor network. In *American control conference* (pp. 2072–2077).
Sussmann, H. J., & Kokotović, P. V. (1991). The peaking phenomenon and the global stabilization of nonlinear systems. *IEEE Transactions on Automatic Control*, *36*(4), 424–440.
Trefethen, L. N., & Bau, D. (1997). *Numerical linear algebra*. SIAM.
Yang, P., Freeman, R., & Lynch, K. (2007). Distributed cooperative active sensing using consensus filters. *Proc. of 2007 IEEE international conference on robotics and automation*, Rome, Italy, (pp. 405–410).
Zavlanos, M. M., & Pappas, G. J. (2005). Controlling connectivity of dynamic graphs. In *IEEE conference on decision and control* (pp. 6388–6393).
Zavlanos, M. M., & Pappas, G. J. (2007). Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, *23*(4), 812–816.
Zhao, F., Shin, J., & Reich, J. (2002). Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, *19*(2), 61–72.



P. Yang received a Ph.D. in Mechanical Engineering from Northwestern University (Evanston, IL) in 2008. He received a B.S. degree in Theoretical and Applied Mechanics from Peking University (Beijing, China) in 2003. His research interests include nonlinear control theory, distributed algorithms, and multi-agent systems.

R.A. Freeman received a Ph.D. in Electrical Engineering from the University of California at Santa Barbara in 1995. Since then he has been a faculty member in the Department of Electrical Engineering and Computer Science at Northwestern University (Evanston, Illinois), where he is currently an Associate Professor. He has been a member of the IEEE Control System Society Conference Editorial Board since 1997, he has served as the Associate Editor of the IEEE Transactions on Automatic Control, and he has served on Program and Operating Committees for the American Control Conference and the IEEE Conference on Decision and Control. His research interests include nonlinear system theory, nonlinear control, robust control, and optimal control. He received the NSF CAREER award in 1997.

G.J. Gordon is an Associate Research Professor in the Department of Machine Learning at Carnegie Mellon University, and the co-Director of the Department's Ph.D. program. He works on multi-robot systems, statistical machine learning, game theory, and planning in probabilistic, adversarial, and general-sum domains. His previous appointments include Visiting Professor at the Stanford Computer Science Department and Principal Scientist at Burning Glass Technologies in San Diego. Dr. Gordon received his B.A. in Computer Science from Cornell University in 1991, and his Ph.D. in Computer Science from Carnegie Mellon University in 1999.

K.M. Lynch received the B.S.E. degree in electrical engineering from Princeton University, Princeton, NJ, in 1989, and the Ph.D. degree in robotics from Carnegie Mellon University, Pittsburgh, PA, in 1996. He is McCormick Professor of Teaching Excellence and Associate Professor of Mechanical Engineering at Northwestern University (Evanston, IL), where he co-directs the Laboratory for Intelligent Mechanical Systems. His research interests include robot manipulation planning, motion planning and control for underactuated mechanical systems, self-organizing multi-agent systems, and human–robot systems. Dr. Lynch received the NSF CAREER Award in 1998 and the IEEE Robotics and Automation Early Academic Career Award in 2001.

S.S. Srinivasa is a Senior Research Scientist with Intel Labs Pittsburgh. He also holds an Adjunct Faculty position at the Robotics Institute at Carnegie Mellon University. He is a co-PI of the Personal Robotics project, in which an anthropomorphic robotic arm and a mobile robot coordinate to accomplish useful manipulation tasks in populated indoor environments. His research focuses on enabling robots to interact faster, better, and smoother with the real world. He received his Ph.D. from the Robotics Institute at Carnegie Mellon University where he developed robust controllers for robotic manipulation. He also has a B.Tech in Mechanical Engineering from the Indian Institute of Technology, Madras.

R. Sukthankar is a senior principal research scientist at Intel Labs Pittsburgh and an adjunct research professor in the Robotics Institute at Carnegie Mellon. He was previously a senior researcher at HP/Compaq's Cambridge Research Lab and a research scientist at Just Research. Rahul received his Ph.D. in Robotics from Carnegie Mellon and his B.S.E. in Computer Science from Princeton. His current research focuses on computer vision and machine learning, particularly in the areas of object recognition and information retrieval.