On Maximizing the Second Smallest Eigenvalue of a State-dependent Graph Laplacian

Yoonsoo Kim and Mehran Mesbahi

***Abstract*— We consider the set $\mathcal{G}$ consisting of graphs of fixed order and weighted edges. The vertex set of graphs in $\mathcal{G}$ will correspond to point masses and the weight for an edge between two vertices is a functional of the distance between them. We pose the problem of finding the best vertex positional configuration in the presence of an additional proximity constraint, in the sense that, the second smallest eigenvalue of the corresponding graph Laplacian is maximized. In many recent applications of algebraic graph theory in systems and control, the second smallest eigenvalue of Laplacian has emerged as a critical parameter that influences the stability and robustness properties of dynamic systems that operate over an information network. Our motivation in the present work is to "assign" this Laplacian eigenvalue when relative positions of various elements dictate the interconnection of the underlying weighted graph. In this venue one would then be able to "synthesize" information graphs that have desirable system theoretic properties.**

***Index Terms*— Networked dynamic systems, graph Laplacian, Euclidean distance matrix, semidefinite programming**

I. Introduction

Consider the set of n mobile elements as vertices of a graph, with the edge set determined by the relative positions between the respective elements. Specifically, we let $\mathcal{G}$ denote the set of graphs of order n with vertex set $V = \{1, 2, \ldots, n\}$ and edge set $E = \{e_{ij},\, i = 1, 2, \ldots, n-1,\, j = 2, \ldots, n;\, i < j\}$ with the weight function

$$w : \mathbf{R}^3 \times \mathbf{R}^3 \to \mathbf{R}_+,$$

assigning to each edge e_{ij}, a function of the distance between the two nodes i and j. Thus we have

$$w_{ij} := w(x_i, x_j) = f(\|x_i - x_j\|), \tag{1}$$

for some $f : \mathbf{R}_+ \to \mathbf{R}_+$, with $x_i \in \mathbf{R}^3$ denoting the position of element i. In our setup the function f in (1) will be required to exhibit a distinct behavior as it traverses the positive real line. For example, we will require that this function assume a constant value of one when the distance between i and j is less than some threshold and then rapidly drop to zero (or some small value) as the distance between these elements increases. Such a requirement parallels the behavior of an information link in a wireless network where the signal power at the receiver side is inversely proportional to the some power of the distance between transmitting and receiving elements [17]. Using this framework, we now consider the configuration problem

$$\mathbf{\Lambda} : \quad \max_x \quad \lambda_2(L_G(x)), \tag{2}$$

where $x := [x_1, x_2, \ldots, x_n]^T \in \mathbf{R}^{3n}$ is the vector of positions for the distributed system, the matrix $L_G(x)$ is a weighted graph Laplacian defined element-wise as

$$[L_G(x)]_{ij} := \begin{cases} -w_{ij} & \text{if } i \neq j, \\ \sum_{s \neq i} w_{is} & \text{if } i = j, \end{cases} \tag{3}$$

and $\lambda_2(L_G(x))$ denotes the second smallest eigenvalue of the state-dependent Laplacian matrix $L_G(x)$ with its spectrum ordered as

$$\lambda_1(L_G) \leq \lambda_2(L_G) \leq \ldots \leq \lambda_n(L_G).$$

Furthermore, we restrict the feasible set of (2) by imposing the proximity constraint

$$d_{ij} := \|x_i - x_j\|^2 \geq \rho_1, \quad \text{for all } i \neq j, \tag{4}$$

preventing the elements from getting arbitrary close to each other in their desire to maximize $\lambda_2(L_G)$ in (2).

The second smallest eigenvalue of the graph Laplacian L_G, also known as the algebraic connectivity of G [7], [13], has emerged as an important parameter in many systems problems defined over networks [6], [11], [14], [16], [19]. In fact, in several recent works [6], [16], [18], it has been observed that $\lambda_2(L_G)$ is a measure of stability and robustness of the networked dynamic system. This observation implies, for example, that small perturbations in the configuration of the networked system will be attenuated back to its equilibrium state(s) with a rate that is proportional to $\lambda_2(L_G)$. When this important graph parameter is considered in a state-dependent setting as proposed in [14], the characterization of a distributed system states that maximize $\lambda_2(L_G)$ emerges as a natural optimization problem. In this venue however, there are only a handful of studies in the literature that are related to such a graph eigenvalue assignment problem (2). In particular, we mention the work of Fallat and Kirkland [5] where a graph-theoretical approach has been proposed to extremize $\lambda_2(L_G)$ over the set of trees of fixed diameter. Also related to the present work are those by Chung and Oden [4] pertaining to bounding the gap between the first two eigenvalues of graph Laplacians, and Berman and Zhang [2] and Guattery and Miller [10], where, respectively, isoperimetric numbers of

February 2005. The research of the authors was supported by National Science Foundation NSF/CMS-0093456.

Y. Kim is currently with the Department of Engineering, University of Leicester, U.K. Email: *yk17@leicester.ac.uk*. M. Mesbahi is with the Department of Aeronautics and Astronautics, University of Washington, Seattle, WA. Email: *mesbahi@aa.washington.edu*

weighted graphs and graph embeddings are employed for lower bounding the second smallest Laplacian eigenvalue. We note that maximizing the second smallest eigenvalue of state dependent graph Laplacians over arbitrary graph constraints is a difficult computational problem [15].

The contribution of this paper is to propose an iterative greedy-type algorithm for problem (2) with a guaranteed local convergence behavior. Although the convergence of this algorithm is provably local in nature, extensive simulations suggest that it often converges to the global maximum when the initial graph is taken to be a path. The outline of the paper is as follows. In §II.A we delineate on the various possible choices for the edge weights for our state-dependent weighted Laplacians. §II.B and §II.C are devoted to the main result of the paper, where an iterative, semidefinite programming-based approach is proposed for the solution of problem $\mathbf{\Lambda}$ (2). A numerical example is then presented in §III followed by some concluding remarks.

A few words on the notation. The 2-norm of vector x will be denoted by $\|x\|$. The spaces of $n \times n$ real matrices and $n \times n$ real symmetric matrices are designated by $\mathbf{R}^{n\times n}$ and $\mathbf{S}^n$, respectively; I_n will be the $n \times n$ identity matrix. The inequalities between symmetric matrices are interpreted in the sense of Löwner ordering, i.e., $A > B$ and $A \geq B$ indicate, respectively, the positive definiteness and positive semi-definiteness of the matrix difference $A - B$.

II. Method

As we mentioned in §I, the general formulation of the problem $\mathbf{\Lambda}$ (2) does not readily hint at being tractable, in the sense of admitting an efficient algorithm for its solution. Generally, maximizing the second smallest eigenvalue of a symmetric matrix subject to matrix inequalities, does not yield to a standard Linear Matrix Inequality approach [3] and subsequently a solution procedure based on interior point methods [1]. The above complication however is alleviated in case of graph Laplacians, where the smallest eigenvalue $\lambda_1(L_G)$ is always zero with the associated eigenvector of $\mathbf{1}$ composed of unit entries. This observation follows directly from the definition (3). Nevertheless, due to the nonlinear dependency of entries of L_G on the relative distance d_{ij} and the presence of constraints (4), the problem $\mathbf{\Lambda}$ (2) assumes the form of a non-convex optimization. In light of this fact, we will proceed to propose an iterative SDP-based approach for this problem. However, before we proceed, we make few remarks on some judicious choices for the function f in (1).

The choice of f in (1) is not only guided by particular applications but also by numerical considerations. A few candidate functions are shown in Figure 1. Although there are a host of choices for f, for our numerical experimentation we have chosen to work with Type-IV functions (the lower right corner in Figure 1), where f assumes the form

$$f(d_{ij}) = \epsilon^{(\rho_1 - d_{ij})/(\rho_1 - \rho_2)}, \quad \epsilon > 0, \tag{5}$$

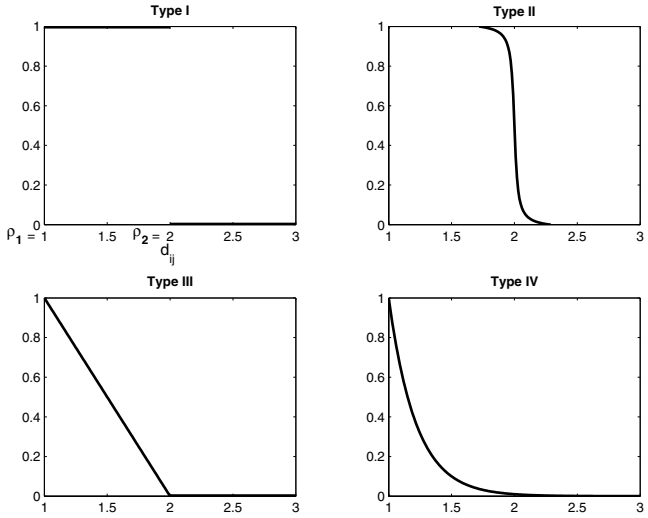

Fig. 1. Several candidates for the function f in (1) where $\rho_1 = 1$ and $\rho_2 = 2$.

given that $d_{ij} \geq \rho_1$. [1] We note that $f(\rho_1) = 1$ and $f(\rho_2) = \epsilon$. Among the advantages of working with functions (5) are their differentiability properties, as well as their ability to capture a situations that is of practical relevance. In many such situations, the strength of an information link is inversely proportional to the relative distance, and decays exponentially after a given threshold (e.g., ρ_2 in (5)) is passed. Furthermore, and possibly more importantly, functions (5) lead to a stable algorithm for our numerical experimentations; a representative set of examples is discussed in §III.

A. Maximizing $\lambda_2(L_G)$

We first present a linear algebraic result in conjunction with the general problem of maximizing the second smallest eigenvalue of graph Laplacians.

Proposition 2.1: Consider the m-dimensional subspace $\mathbf{P} \subseteq \mathbf{R}^n$ spanned by the vectors $p_i \in \mathbf{R}^n$, $i = 1, \ldots, m$. Denote $P := [p_1, \ldots, p_m] \in \mathbf{R}^{n\times m}$. Then for $M \in \mathbf{S}^n$ one has

$$x^T M x > 0 \quad \text{for all nonzero } x \in \mathbf{P}$$

if and only if

$$P^T M P > 0. \tag{6}$$

Proof: An arbitrary nonzero element $x \in \mathbf{P}$ can be written as

$$x = \alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_m p_m$$

for some $\alpha_1, \ldots, \alpha_m \in \mathbf{R}$, not all zeros, and thus $x = Py$, where $y := [\alpha_1, \alpha_2, \ldots, \alpha_m]^T$. Consequently the first

[1] We have also used functions of the form $(1/d_{ij})^\gamma$, where γ is a positive number and $f(\rho_2) = \epsilon$. Our simulation results in §III have turned out to be exactly the same for these functions as compared with those obtained using functions of the form (5).

inequality in (6) is equivalent to

$$(Py)^T M(Py) = y^T P^T MPy > 0$$

for all nonzero $y \in \mathbf{R}^m$, or in other words, having $P^T MP > 0$; we note that $P^T MP \in \mathbf{S}^m$. ■

Corollary 2.2: For a graph Laplacian L_G the constraint

$$\lambda_2(L_G) > 0, \tag{7}$$

is equivalent to

$$P^T L_G P > 0,$$

where $P = [p_1, p_2, \ldots, p_{n-1}]$, and the unit vectors $p_i \in \mathbf{R}^n$ are chosen such that

$$p_i^T \mathbf{1} = 0 \quad (i = 1, 2, \ldots, n-1)$$

and

$$p_i^T p_j = 0 \quad (i \neq j). \tag{8}$$

Proof: It is well-known that for $G \in \mathcal{G}$,

$$L_G \geq 0 \quad \text{and} \quad L_G \, \mathbf{1} = 0, \tag{9}$$

and thereby, the smallest eigenvalue of L_G is always zero and **rank** $L_G \leq n-1$. This implies that (7) is equivalent to having

$$x^T L_G x > 0 \quad \text{for all nonzero } x \in \mathbf{1}^\perp, \tag{10}$$

where

$$\mathbf{1}^\perp := \{x \in \mathbf{R}^n \mid \mathbf{1}^T x = 0\}.$$

In view of Proposition 2.1, the condition (10) is equivalent to having $P^T L_G P > 0$, with P denoting the matrix of vectors spanning the subspace $\mathbf{1}^\perp$. Without loss of generality, this subspace can be identified with the basis unit vectors satisfying (8). ■

In view of Corollary 2.2, the problem $\mathbf{\Lambda}$ (2) can be re-stated as:

$$\begin{aligned} \mathbf{\Lambda}: \quad & \max_x \quad \gamma && (11)\\ \text{s.t.} \quad & d_{ij} := \|x_i - x_j\|^2 \geq \rho_1, && (12)\\ & P^T L_G(x) P \geq \gamma \, I_{n-1}, && (13) \end{aligned}$$

where $i = 1, 2, \ldots, n-1, j = 2, \ldots, n$, $i < j$, and the unit vectors p_i's in (8) form the columns of the matrix P.

B. Discrete and greedy

We now proceed to view the problem $\mathbf{\Lambda}$ (2) in an iterative setting, where the goal is shifted toward finding an algorithm that attempts to maximize the second smallest eigenvalue of the graph Laplacian at each step. Toward this aim, we first differentiate both sides of (12) with respect to time as

$$2\{\dot{x}_i(t) - \dot{x}_j(t)\}^T \{x_i(t) - x_j(t)\} = \dot{d}_{ij}(t), \tag{14}$$

and then employ Euler's first discretization method, with Δt as the sampling time,

$$x(t) \to x(k), \quad \dot{x}(t) \quad \to \quad \frac{x(k+1) - x(k)}{\Delta t},$$

to rewrite (12) as

$$\begin{aligned} 2\{x_i(k+1) - x_j(k+1)\}^T \{x_i(k) - x_j(k)\} \\ = d_{ij}(k+1) + d_{ij}(k). \end{aligned}$$

Similarly, the state dependent Laplacian $L_G(x)$ in (13) is discretized by first differentiating the terms w_{ij} with respect to time, and then having

$$\begin{aligned} w_{ij}(k+1) \quad &= \quad w_{ij}(k) \\ &- \quad \epsilon^{(\rho_1 - d_{ij}(k))/(\rho_1 - \rho_2)} \{d_{ij}(k+1) - d_{ij}(k)\}; \end{aligned}$$

recall that we are employing functions of the form (5) in (1). The discrete version of the state dependent Laplacian, $L_G(k)$ now assumes the form

$$[L_G(k)]_{ij} = \begin{cases} -w_{ij}(k) & \text{if } i \neq j, \\ \sum_{s \neq i} w_{is}(k) & \text{if } i = j. \end{cases}$$

Putting it all together, we arrive at the iterative step of solving the optimization problem

$$\begin{aligned} \mathbf{\Lambda}_k: \quad & \max_{x(k+1)} \quad \gamma && (15)\\ \text{s.t.} \quad & 2\{x_i(k+1) - x_j(k+1)\}^T \{x_i(k) - x_j(k)\} \\ & = d_{ij}(k+1) + d_{ij}(k), && (16)\\ & d_{ij}(k+1) \geq \rho_1, && (17)\\ & P^T L_G(k+1) P \geq \gamma \, I_{n-1}, && (18) \end{aligned}$$

for $i = 1, 2, \ldots, n-1, j = 2, \ldots, n, i < j$, and $x(k) := [x_1(k), x_2(k), \ldots, x_n(k)]^T \in \mathbf{R}^{3n}$. Thereby, the algorithm is initiated at time $k = 0$ with an initial graph (configuration) G_0, and then for $k = 0, 1, 2, \ldots$, we proceed to iteratively find graph that maximize $\lambda_2(L_G(k+1))$. This greedy procedure is then iterated upon until the value of $\lambda_2(L_G(k))$ can not be improved further. We note that the proposed greedy algorithm converges, as the sequence generated by it is nondecreasing and bounded from above.[2]

C. Further Considerations

In previous section, we proposed an algorithm that converges to a local optimal vertex positional configuration, in terms of maximizing the quantity $\lambda_2(L_G)$. However, by replacing the non-convex constraint (12) with its linear approximation (16)-(17), one introduces a potential inconsistency between the position and the distance vectors. In this section, we provide two remedies to avoid such potential complications. Let us first recall the notion of Euclidean Distance Matrix (EDM). Given the position vectors $x_1, x_2, \ldots, x_n \in \mathbf{R}^3$, the EDM $D = [d_{ij}] \in \mathbf{R}^{n \times n}$ is defined entry-wise as

$$[D]_{ij} = d_{ij} = \|x_i - x_j\|^2 \quad \text{for } i, j = 1, 2, \ldots, n.$$

The EDM matrices are nicely characterized in terms of linear matrix inequalities [9].

[2]The second smallest eigenvalue of L_G for an n vertex graph is bounded by n, corresponding to the complete graph K_n [8].

Theorem 2.3: A matrix $D = [d_{ij}] \in \mathbf{R}^{n\times n}$ is an EDM if and only if

$$JDJ \geq 0, \quad (19)$$

$$d_{ii} = 0 \quad \text{for } i = 1, 2, \ldots, n, \quad (20)$$

where $J := I - \mathbf{1}\mathbf{1}^T/n$.

Theorem 2.3 allows us to guarantee that by adding the two convex constraints (19)-(20) to problem $\mathbf{\Lambda}_k$ (15)-(18), we always obtain consistency among the position and distance variables at each iteration step. Moreover, by updating the values of $d_{ij}(k)$'s and $[L(k)]_{ij}$'s in (16) and (18) *after* calculating the values of $x(k)$, we can further reduce the effect of linearization in the proposed procedure. To further expand on this last point, suppose that $x_1(k), x_2(k), \ldots, x_n(k)$, $d_{ij}(k)$'s and $[L(k)]_{ij}$, $i = 1, 2, \ldots, n-1, j = 2, \ldots, n, i < j$, have been obtained after solving the problem $\mathbf{\Lambda}_k$ (15)-(18). Our proposed modification to the original algorithm thus amounts to updating the values of $d_{ij}(k)$ and $[L(k)]_{ij}$, based on the computed values of $x_1(k), x_2(k), \ldots, x_n(k)$, before initiating the next iteration.

III. Simulation Results

For our simulations we used SeDuMi [1] to solve the required semidefinite programs. Figure 2 depicts the behavior of six mobile elements under the guidance of the proposed algorithm, leading to a planar configuration that locally maximizes $\lambda_2(L_G)$. The constants ϵ, ρ_1, and ρ_2 in (5) are chosen to be 0.1, 1, and 1.5, respectively. The algorithm was initialized with a configuration that corresponds to a path. The sequence of configurations thereafter converges to the truss-shape graph with the $\lambda_2(L_G)$ of 1.6974. For these set of parameters, the truss-shape graph as suggested by the algorithm is the global maximum over the set of graphs on six vertices that can be configured in $\mathbf{R}^2$.[3] Using the same simulation scenario, but this time, in search of an optimal positional configuration in $\mathbf{R}^3$, the algorithm leads to the trajectories shown in Figure 3. In this case, the graph sequence converges to an octahedron-shape configuration with $\lambda_2(L_G) = 4.02$.

Increasing the number of nodes to eight, the algorithm was initialized as the unit cube; the resulting trajectories are shown in Fig 4. In this figure, the edges between vertices i and j indicate that $d_{ij} \leq \rho_2 = 1.5$. The solid lines in Figure 4 represent the final configuration with $\lambda_2(L_G) = 2.7658$. Once again, an exhaustive search procedure indicates that the proposed algorithm does lead to the global optimal configuration (see Table 1). We like to remark however that the choice for the function f in (5) and the initial configuration, are critical to the performance of the proposed algorithm. For example, when this function is chosen to be of Type-I in Figure 1 and the initial graph as a disconnected graph, the algorithm terminates right after initialization, as any small perturbation on the initial graph does not lead to an improvement in the value of $\lambda_2(L_G)$. Choosing a Type-IV function in Figure 1 on the other hand, always lead to a connected configuration with a positive $\lambda_2(L_G)$, even when the algorithm is initialized via a disconnected graph.

[3] A global maximum may be found in the following exhaustive manner: first, define a space large enough guaranteed to contain the optimal configuration. Then grid this region and search over the set of all n grid points for the configuration that leads to maximum $\lambda_2(L_G)$.

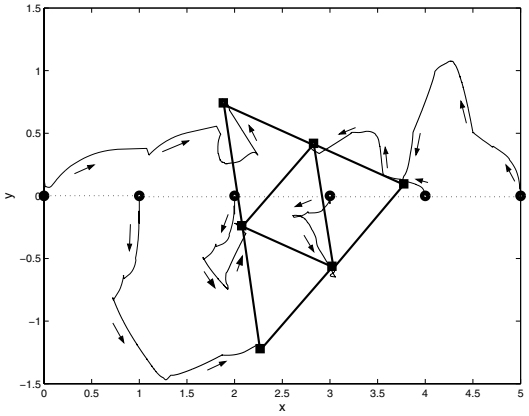

Fig. 2. The trajectory generated by the proposed algorithm for 6 nodes in $\mathbf{R}^2$: the configuration evolves from path (circles) to truss (squares).

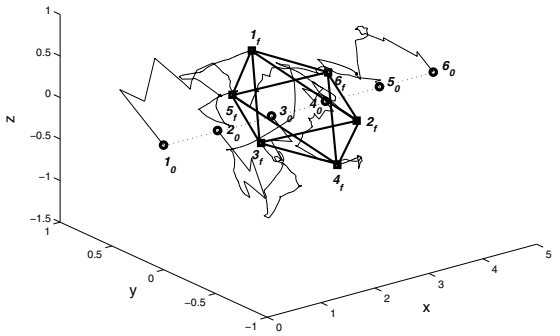

Fig. 3. The trajectory generated by the proposed algorithm for six nodes in $\mathbf{R}^3$: the configuration evolves from path (circles, $1_0, \ldots, 6_0$) to octahedron (squares, $1_f, \ldots, 6_f$).

# of nodes in $\mathbf{R}^n$	$\lambda_2(L_G)$	$\lambda_2(L_{\widetilde{G}})$	$\lambda_2(L_G)/\lambda_2(L_{\widetilde{G}})$
6 nodes in $\mathbf{R}^2$	1.6974	1.6972	1.001
6 nodes in $\mathbf{R}^3$	4.02	4.0	1.005
8 nodes in $\mathbf{R}^3$	2.7658	2.7639	1.007

TABLE I

Comparing the λ_2 values for the type-IV weighted graph G as realized by the algorithm and those corresponding to the associated $0-1$ weighted graph $\widetilde{G}$.

IV. Concluding Remarks

We considered the problem of maximizing the second smallest eigenvalues of a state-dependent graph Laplacian. This problem is of importance when the positions of a set of

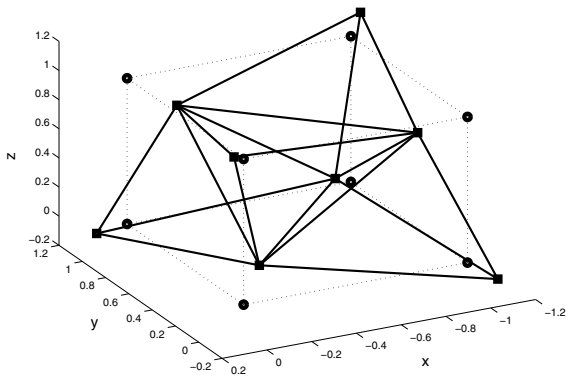

Fig. 4. Evolution of the proposed algorithm for 8 nodes in $\mathbf{R}^3$: the configuration evolves from 3-cube (circles) to octahedron (squares).

dynamic elements- operating over an information network- can be chosen for robust system performance. We proposed an iterative algorithm for this problem that employs a semidefinite programming solver at each recursive step. Although the algorithm has a local convergence behavior, extensive simulations suggest that it often leads to a globally optimal positional configuration.

V. ACKNOWLEDGMENTS

The authors gratefully acknowledge suggestions and comments by the anonymous reviewers.

REFERENCES

[1] *http://fewcal.kub.nl/sturm/software/sedumi.html*

[2] A. Berman and X-D. Zhang. Lower bounds for the eigenvalues of Laplacian matrices, *Linear Algebra and its Applications*, 316: 13-20, 2000.

[3] S. Boyd and L. Vandenberghe. *Convex Programming*, Cambridge University Press, 2003.

[4] F. R. K. Chung and K. Oden. Weighted graph Laplacians and isoperimetric inequalities. *Pacific Journal of Mathematics*, 192 (2): 257-273, 2000.

[5] S. Fallat and S. Kirkland. Extremizing algebraic connectivity subject to graph theoretic constraints, *The Electronic Journal of Linear Algebra*, (3) 1: 48–74, 1998.

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

[7] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its applications in graph theory. *Czechoslovak Mathematical Journal*, (26) 100: 619-633, 1975.

[8] C. Godsil and G. Royle. *Algebraic Graph Theory*, Springer-Verlag, 2001.

[9] J. Gower. Properties of Euclidean and non-Euclidean distance matrices, *Linear Algebra and its Applications*, (67) 1: 81–97, 1985.

[10] S. Guattery and G. L. Miller. On the quality of spectral separators, *SIAM Journal on Matrix Analysis and Applications*, 19 (3): 701-719, 1998.

[11] 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, 2003.

[12] Y. Kim and M. Mesbahi. Quadratically constrained attitude control via semidefinite programming, *IEEE Transaction on Automatic Control* (49) 5: 731-735, 2004.

[13] R. Merris. Laplacian matrices of graphs: a survey, *Linear Algebra and its Applications,* 197 (1): 143–176, 1994.

[14] M. Mesbahi. State-dependent graphs and their controllability properties, *IEEE Transaction on Automatic Control* (to appear).

[15] H. Q. Ngo and D.-Z. Du. Notes on the complexity of switching networks, in *Advances in Switching Networks*, H. Q. Ngo and D.-Z. Du (Editors), Kluwer Academic Publishers, pp. 307-357, 2000.

[16] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays, *IEEE Transactions on Automatic Control*, (49) 9: 1520- 1533, 2004.

[17] K. Pahlavan and A. H. Levesque. *Wireless Information Networks*, John Wiley and Sons, Inc., New York, 1995.

[18] H. Tanner, A. Jadbabaie, and G. Pappas. "Flocking in fixed and switching networks," *Automatica* (submitted).

[19] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging, *Systems and Control Letters*, 53: 65-78, 2004.