Designing Node and Edge Weights of a Graph to Meet Laplacian Eigenvalue Constraints

S. Yusef Shafi, Murat Arcak, Laurent El Ghaoui
University of California, Berkeley
{yusef,arcak,elghaoui}@eecs.berkeley.edu

***Abstract*—We consider agents connected over a network, and propose a method to design an optimal interconnection such that the gap between the largest and smallest Laplacian eigenvalues of the graph representing the network is minimized. We study ways of imposing constraints that may arise in physical systems, such as enforcing lower bounds on connectivity and upper bounds on gain as well as network cost. In particular, we show that node and edge weights of a given graph can be simultaneously adjusted via convex optimization to achieve improvements in its Laplacian spectrum.**

I. Introduction

A well-studied tool for characterizing the interconnection topology of a network is the graph Laplacian matrix [1]. In particular, the spectrum of the Laplacian contains useful information about the dynamics of the interconnected system. For example, the smallest positive eigenvalue of a Laplacian, known as the algebraic connectivity, or Fiedler eigenvalue [2], is a measure of how well connected a network is. The Fiedler eigenvalue has been extensively studied in diverse application areas, including motion coordination [3][4][5] and spectral clustering in image segmentation [6]. In motion coordination, the Fiedler eigenvalue determines bounds on convergence speed, while in spectral clustering, the eigenvector corresponding to the Fiedler eigenvalue is used to bipartition a graph representation of an image. On the other hand, the largest eigenvalue of the Laplacian affects stability of the interconnected system. Indeed, the largest eigenvalue must be sufficiently small for stability of discrete-time consensus algorithms [4][7] and for continuous-time formation control algorithms when the agent dynamics are prone to high-gain instability[8].

We present an optimization scheme to meet a lower bound constraint on the Fiedler eigenvalue and an upper bound constraint on the largest eigenvalue. The problem is divided into two increments: one consisting of deriving a feasibility program that incorporates eigenvalue constraints via edge and node weighting, and the other consisting of implementing various objectives within the framework. A general scheme through which both edges and nodes can be adjusted simultaneously is discussed, as well as more restrictive scenarios where either edge weights or node weights are specified. Our approach is unique because it deals with the assignment of weights to both edges and nodes, and we show that the process can be done separately or simultaneously.

Convex optimization solutions to several graph problems are well-documented in the literature, including iterations for fast distributed linear averaging (FDLA) [9], minimizing total effective resistance on a graph [10], fastest mixing Markov chains [11] and processes [12], and Fiedler eigenvalue maximization through vertex positioning [13]. In FDLA, given a particular interconnection structure for a discrete system with symmetric interconnection topology, the number of iterations required for linear averaging is minimized by finding a particular weight distribution that assigns optimal update laws for each node's state based on a weighted combination of its current value and that of its neighbors. The goal in network resistance problems is to minimize the total effective resistance on a graph by assigning different weights representing resistances to the links connecting the nodes of an electrical network. Given a graph structure, the aim for fastest mixing Markov chains and processes is to find the optimal transition probabilities between states to reach a stationary distribution as quickly as possible. Finally, vertex positioning aims to find the optimal locations of vertices in order to assign edge weights to maximize the Fiedler eigenvalue. We go beyond what has been done by addressing the Fiedler and largest eigenvalues simultaneously, and in so doing, control the gap between them. Furthermore, we show how node and edge weighting can be simultaneously manipulated through convex optimization to yield a more desirable Laplacian spectrum.

The remainder of the paper is organized as follows. Section II outlines some basic definitions and results from linear algebra and graph theory. Section III outlines a general optimization framework that allows us to develop feasibility constraints for network design based on edge and node weighting. Section IV elaborates on different convex and quasiconvex objectives enabled by the framework of Section III. Section V presents numerical examples, and Section VI concludes with a discussion of future research directions.

II. Preliminaries

We review the following results from Linear Algebra to facilitate the subsequent discussion. The first result for the

eigenvalues of a product of two matrices is well-known ([14], Thm. 1.3.20):

Lemma 2.1: Let $A \in R^{n\times m}$, $B \in R^{m\times n}$, and $n \geq m$. Then AB and BA have m identical eigenvalues with AB having $n-m$ additional eigenvalues at zero.

Definition 2.2: The square matrices A and B are **congruent** if $B = SAS^T$ for some square, nonsingular S.

The following lemma is more commonly known as Sylvester's Law of Inertia ([14], Thm. 4.5.8):

Lemma 2.3: Let A, $B \in R^{n\times n}$ be symmetric matrices. A and B are congruent if and only if A and B have the same inertia, i.e., the same number of positive, negative, and zero eigenvalues.

Finally, an inequality due to Sylvester characterizes the relationship between the eigenvalues of two matrices and their products ([15], Section 3.5):

Lemma 2.4: Given two matrices $A \in R^{m\times n}$ and $B \in R^{n\times p}$, the following inequality holds:

$$\begin{aligned} \text{rank}(A) + \text{rank}(B) - n \leq \text{rank}(AB) \\ \leq \min\{\text{rank}(A), \text{rank}(B)\}. \end{aligned} \tag{1}$$

We define a graph $G = G(V, E)$ to be a collection of nodes V connected by the set of edges E. Throughout, we assume that v_i is linked to v_j by an edge if and only if v_j is linked to v_i by an edge, i.e., the graph is *undirected*. A graph is *connected* if there exists a sequence of edges linking any two vertices v_i and v_j. Given a graph with n nodes and m edges, the graph incidence matrix A is an $n \times m$ matrix, each of whose columns k represent an edge linking node v_i and v_j with $[A]_{ik} = 1$, $[A]_{jk} = -1$, and $[A]_{lk} = 0$ for all $l \neq i, j$.

We define the *node- and edge- weighted graph Laplacian* (henceforth *Laplacian*) as:

$$L_g = Y^{-1}A\Lambda A^T \tag{2}$$

where Y and Λ are positive diagonal matrices representing the weights assigned to the nodes and edges of the graph, respectively, and A is the incidence matrix. In our derivations we denote by L the *edge-weighted Laplacian*:

$$L = A\Lambda A^T. \tag{3}$$

Several key facts about the Laplacian follow [1]. First, L is symmetric positive semidefinite, with at least one eigenvalue at zero, with corresponding eigenvector $\frac{1}{\sqrt{n}}[1\ldots 1]^T$ (henceforth written $\frac{1}{\sqrt{n}}\mathbf{1}$). If the graph is connected, it has exactly one eigenvalue at zero. The spectrum of Laplacian describes the dynamics of the underlying network. The most important eigenvalue in dictating connectivity properties of the graph is the second smallest eigenvalue, known as Fiedler eigenvalue [2]. The next lemma shows how the spectral properties of L carry over to L_g.

Lemma 2.5: L_g has exclusively real, nonnegative eigenvalues. If the graph L_g represents is connected, then all eigenvalues are positive except for one at zero.

Proof: A similarity transformation brings L_g to the symmetric form $Y^{-1/2}LY^{-1/2}$, and so L_g has real eigenvalues. Furthermore, the symmetric matrices $Y^{-1/2}LY^{-1/2}$ and L are congruent, and so Lemma 2.3 guarantees that L_g has no negative eigenvalues. When L represents a connected graph and thus has only one eigenvalue at zero, all eigenvalues of L_g excepting one at zero are positive. ■

III. Upper and Lower Eigenvalue Constraints

We derive a constraint set that defines a feasible region whereby through edge and node weightings, a weighted Laplacian satisfies specified spectral constraints.

Our goal is to find feasible Y and Λ such that L_g has all $n-1$ nonzero eigenvalues between λ and $\kappa\lambda$, where κ is to be minimized and λ is a prescribed parameter.

A. The Upper Eigenvalue Constraint

The upper eigenvalue requirement is that no eigenvalue of L_g may be larger than $\kappa\lambda$. Then Y, Λ must be chosen so $\kappa\lambda I - Y^{-1}L$ has only nonnegative eigenvalues.

To derive a matrix inequality constraint, note that a similarity transformation brings $\kappa\lambda I - Y^{-1}L$ to the symmetric form $\kappa\lambda I - Y^{-1/2}LY^{-1/2}$. This matrix is in turn congruent to $\kappa\lambda Y - L$, and so, by Lemma 2.3, the matrices' eigenvalues share the same set of signs. Therefore, the matrix inequality

$$\kappa\lambda Y - L \succeq 0 \tag{4}$$

enforces the upper bound constraint.

B. The Lower Eigenvalue Constraint

The lower eigenvalue requirement is that no eigenvalue λ_i for $i = 2, \ldots, n$ of L_g may be less than λ. We begin by using Gram-Schmidt orthogonalization to construct an orthonormal matrix of basis vectors G starting from the basis

$$\left\{\frac{1}{\sqrt{n}}\mathbf{1}, e_2, \ldots, e_n\right\}. \tag{5}$$

From the resulting basis, form an $n \times (n-1)$ matrix Q consisting of the vectors orthogonal to $\frac{1}{\sqrt{n}}\mathbf{1}$. The matrix

QQ^T is then a projection matrix onto the orthogonal complement of $\frac{1}{\sqrt{n}}\mathbf{1}$. It follows that

$$QQ^T = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T \tag{6}$$

$$Q^TQ = I. \tag{7}$$

Since L is orthogonal to $\frac{1}{\sqrt{n}}\mathbf{1}$,

$$LQQ^T = L. \tag{8}$$

We first show that the lower bound constraint is equivalent to $Y^{-1}L - \lambda QQ^T$ having no negative eigenvalues.

Lemma 3.1: Let Q be the matrix above whose columns are the orthonormal vectors to $\frac{1}{\sqrt{n}}\mathbf{1}^T$. Then for a connected graph, $Y^{-1}L - \lambda QQ^T$ shifts all positive eigenvalues of L_g to the left by λ and leaves unchanged the eigenvalue at zero.

Proof: First note from (8) that

$$M_1 = Y^{-1}L - \lambda QQ^T \tag{9}$$

$$= Y^{-1}LQQ^T - \lambda QQ^T \tag{10}$$

$$= (Y^{-1}LQ - \lambda Q)Q^T \tag{11}$$

Lemma 2.1 shows that this matrix has the same set of eigenvalues as

$$M_2 = Q^T(Y^{-1}LQ - \lambda Q) \tag{12}$$

$$= Q^TY^{-1}LQ - \lambda I \tag{13}$$

with M_1 having an additional eigenvalue at zero. Similarly, $Q^TY^{-1}LQ$ has the same eigenvalues as $Y^{-1}L$, excluding the latter's eigenvalue at zero. Thus, M_2 shifts the nonzero eigenvalues of $Y^{-1}L$ to the left by λ. Using the relationship between the eigenvalues of M_1 and M_2, we conclude that M_1 shifts the nonzero eigenvalues of L_g to the left by λ and leaves the zero eigenvalue unchanged. ■

We now prove an intermediate lemma that will allow us to derive a convex relaxation to the lower bound constraint.

Lemma 3.2: Let Q be as above and D be a symmetric matrix. If $Q^TDQ \succeq 0$, then D has at most one negative eigenvalue.

Proof: Since D is symmetric, it is diagonalizable by a unitary transformation. Let W and S be the unitary and diagonal matrices of eigenvectors and eigenvalues of D, respectively, such that $D = W^TSW$. We will proceed by contradiction, assuming that D has two or more negative eigenvalues.

We begin by writing the matrix equivalences

$$\begin{aligned} Q^TDQ &= Q^TW^TSWQ \\ &= (WQ)^TS(WQ) \\ &\succeq 0. \end{aligned} \tag{14}$$

It is easy to see that the $n \times (n-1)$ matrix product WQ has full column rank by an application of Sylvester's Inequality (Lemma 2.4):

$$\begin{aligned} \text{rank}(W) + \text{rank}(Q) - n &\leq \text{rank}(WQ) \\ n - 1 &\leq \text{rank}(WQ) \\ \Rightarrow \text{rank}(WQ) &= n - 1. \end{aligned} \tag{15}$$

Let z be any vector contained in the range space of WQ. Then

$$z^TSz \geq 0. \tag{16}$$

Now let u_i and u_j be unit eigenvectors of S corresponding to two negative eigenvalues of S. Note that u_i and u_j are also canonical standard basis vectors of R^n. Since the dimension of the domain of D is n, that of Q is $n-1$, and that of $\text{span}\{u_i, u_j\}$ is two, it must be the case that $\{\text{range}(Q) \cap \text{span}\{u_i, u_j\}\}$ is nonempty. Then for any $z \in \{\text{range}(Q) \cap \text{span}\{u_i, u_j\}\}$, the quadratic form of equation (16) is negative, a contradiction. ■

Theorem 3.3: Let L be a symmetric matrix, Y be a positive diagonal matrix, and Q be as above. The constraint

$$Q^T(L - Y)Q \succeq 0 \tag{17}$$

implies $Y^{-1}L - QQ^T$ has only nonnegative eigenvalues.

Proof: First note that the matrices

$$M_3 = Y^{-1}L - I \tag{18}$$

$$M_4 = Y^{-1/2}LY^{-1/2} - I \tag{19}$$

are related by a similarity transformation, and so they share the same spectrum. The matrix M_4 is symmetric, and congruent to the symmetric matrix

$$M_5 = L - Y. \tag{20}$$

Thus, Lemma 2.3 implies that the eigenvalues of M_3 and M_5 share the same set of signs.

That M_3 has at least one negative eigenvalue follows immediately since M_3 must have an eigenvalue at -1 corresponding to the eigenvector $\frac{1}{\sqrt{n}}\mathbf{1}^T$. Therefore, M_5 has at least one negative eigenvalue. Applying Lemma 3.2 in light of (17) guarantees that M_5 has at most one negative eigenvalue. Thus, M_3 has exactly one negative eigenvalue that must be at -1 with corresponding eigenvector $\frac{1}{\sqrt{n}}\mathbf{1}^T$.

Lemma 3.1 implies that $M_1 = Y^{-1}L - QQ^T$ and M_3 share the same spectrum, except for the latter's eigenvalue at -1 being replaced by an eigenvalue at zero. Thus,

$$Q^T(L - Y)Q \succeq 0 \tag{21}$$

implies $Y^{-1}L - QQ^T$ has only nonnegative eigenvalues. ■

Corollary 3.4: Let Y be a positive diagonal matrix and λ be a positive scalar. Then

$$Q^T(L - \lambda Y)Q \succeq 0 \tag{22}$$

implies $Y^{-1}L - \lambda QQ^T$ has only nonnegative eigenvalues.

Proof: Define $\tilde{Y} = \lambda Y$. Note that $\tilde{Y}^{-1} = \frac{1}{\lambda}Y^{-1}$. The corollary follows directly with Y in Theorem 3.3 replaced by $\tilde{Y}$. ■

C. The Feasibility Program

The lower and upper eigenvalue constraints result in the following feasibility program, where κ is a design parameter and ϵ is a positive constant.

$$\begin{aligned} \text{find} \quad & Y,\ \Lambda \\ \text{subject to} \quad & \kappa\lambda Y - A\Lambda A^T \succeq 0 \\ & Q^T(A\Lambda A^T - \lambda Y)Q \succeq 0 \\ & Y \succeq \epsilon I \\ & \Lambda \succeq 0 \\ & \Lambda,\ Y \text{ diagonal} \end{aligned} \tag{23}$$

IV. Incorporating Objectives and Other Constraints

The basic feasibility program derived above can be endowed with several choices of objectives and design constraints. We may specify Fiedler and maximum eigenvalues, and either Λ or Y, and then, find a feasible Y or Λ, respectively. Alternatively, we may specify only one of the two eigenvalues, and find the matrix Y or Λ that results in the smallest maximum eigenvalue or largest Fiedler eigenvalue. The problem can also be posed more generally, with neither Y nor Λ specified. As another consideration, one could impose cost constraints on the weightings. We derive several optimization models and provide illustrative examples in the following discussion.

A. Edge Weighting

In edge weighting problems, it is assumed that node weights are fixed; i.e., Y is given.

The goal in the first problem is to minimize the largest eigenvalue $\kappa\lambda$ given a specified Fiedler eigenvalue λ:

$$\begin{aligned} \text{minimize} \quad & \kappa \\ \text{subject to} \quad & \kappa\lambda Y - A\Lambda A^T \succeq 0 \\ & Q^T(A\Lambda A^T - \lambda Y)Q \succeq 0 \\ & \Lambda \succeq 0 \\ & \Lambda \text{ diagonal.} \end{aligned} \tag{24}$$

The goal in the second problem is to maximize the Fiedler eigenvalue. We denote by λ the largest eigenvalue, and represent the Fiedler eigenvalue λ_2 by $\frac{1}{\kappa}\lambda$. We define $k = -\frac{1}{\kappa}$:

$$\begin{aligned} \text{minimize} \quad & k \\ \text{subject to} \quad & \lambda Y - A\Lambda A^T \succeq 0 \\ & Q^T(A\Lambda A^T + k\lambda Y)Q \succeq 0 \\ & \Lambda \succeq 0 \\ & \Lambda \text{ diagonal.} \end{aligned} \tag{25}$$

Both (25) and (26) are convex semidefinite programs.

B. Node Weighting

We now assume that edge weights are fixed; i.e., Λ is given. The same problems as were considered for edge weighting are considered for node weighting. Once again, they are both convex semidefinite programs:

$$\begin{aligned} \text{minimize} \quad & k \\ \text{subject to} \quad & \lambda Y + kA\Lambda A^T \succeq 0 \\ & Q^T(A\Lambda A^T - \lambda Y)Q \succeq 0 \\ & Y \succeq \epsilon I \\ & Y \text{ diagonal} \end{aligned} \tag{26}$$

$$\begin{aligned} \text{minimize} \quad & \kappa \\ \text{subject to} \quad & \lambda Y - A\Lambda A^T \succeq 0 \\ & Q^T(\kappa A\Lambda A^T - \lambda Y)Q \succeq 0 \\ & Y \succeq \epsilon I \\ & Y \text{ diagonal.} \end{aligned} \tag{27}$$

C. Simultaneous Optimization

Simultaneous design of Y and Λ affords the opportunity to achieve smaller gaps between λ_2 and λ_{max} than possible via either edge or node minimization alone. We note that the variable ϵ is a design choice that affects scaling of the solutions Y and Λ. Corollary 3.4 implies that any positive scaling can be chosen while still preserving feasibility, and in the case of (29), the objective as well. A choice in ϵ is useful when wanting to find physically implementable Y and Λ. The first problem we consider is again minimizing the largest eigenvalue given a specified Fiedler eigenvalue. The problem is the following.

$$\begin{array}{ll} \text{minimize} & k \\ \text{subject to} & \lambda Y + kA\Lambda A^T \succeq 0 \\ & Q^T A\Lambda A^T Q - \lambda Q^T Y Q \succeq 0 \\ & Y \succeq \epsilon I \\ & \Lambda \succeq 0 \\ & \Lambda,\, Y \text{ diagonal} \end{array} \tag{28}$$

This problem is quasiconvex, and thus can still be solved efficiently. Define the nominal Laplacian L_0 as

$$L_0 = AA^T. \tag{29}$$

We denote by λ_2 and λ_N the Fielder and largest eigenvalues of L_0, respectively. With choices of k coming from a bisection on $\left[\frac{\lambda_2}{\lambda_N}, 1\right]$, the program is convex for each k and amounts to a feasibility problem.

Given an eigenvalue separation requirement κ such that $\sigma\{Y^{-1}A\Lambda A^T\} \in [\lambda, \kappa\lambda]$ for all nonzero eigenvalues, an alternative objective is to minimize trace(Λ).

$$\begin{array}{ll} \text{minimize} & \text{trace}(\Lambda) \\ \text{subject to} & \kappa\lambda Y - A\Lambda A^T \succeq 0 \\ & Q^T A\Lambda A^T Q - \lambda Q^T Y Q \succeq 0 \\ & Y \succeq \epsilon I \\ & \Lambda \succeq 0 \\ & \Lambda,\, Y \text{ diagonal} \end{array} \tag{30}$$

V. Numerical Examples

The edge and node weighting programs outlined above were run for three different graphs. The objective is to minimize the *conditioning ratio*, that is, $\frac{\lambda_n}{\lambda_2}$. The design variable in the first two examples is the Fiedler eigenvalue λ_2, while the design variable in the third example is the largest eigenvalue λ_n. In all cases, there is assumed to be no regard to the cost of edge and node weighting schemes. We specify the design variable $\epsilon = 0.1$. As shown below, the experiments running the simultaneous program produced improved results over the independent edge and node weighting programs. All experiments were implemented using the CVX software package[16].

A. Twenty Node Chain Graph

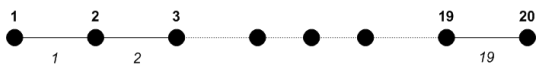

Fig. 1. Twenty Node Chain Graph

The Fiedler eigenvalue of the nominal Laplacian for a twenty node chain graph (Figure 1) is

$$\lambda_2 = 0.0246, \tag{31}$$

and the conditioning ratio is 161.6016.

First, the edge and node weighting SDPs (25) and (27) were run independently. The desired lower eigenvalue was set to λ_2. The edge weighting SDP by itself produced no re-weighting of edges different from identity, and so the conditioning ratio was unchanged. In contrast, the node weighting SDP resulted in a symmetric pattern indicating growth at the edges and center. Most of the nodes were weighted at the same value. The conditioning ratio was somewhat improved at 123.6423.

The simultaneous optimization (29), on the other hand, produced a marked improvement in the conditioning ratio at 53.0407. The edge weighting is no longer flat, and the node weighting profile is further accentuated as compared to that of the solution of (27). The profiles are shown in Figures 2 and 3.

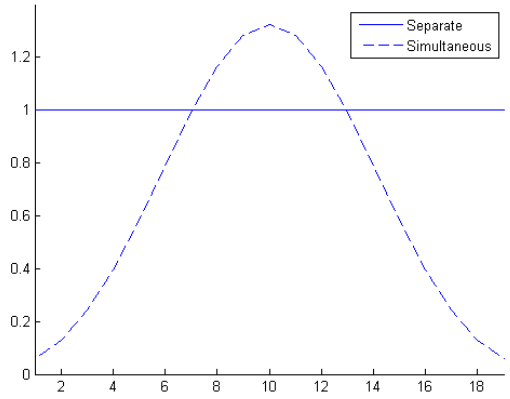

Fig. 2. Edge Weight Profiles for Chain Graph

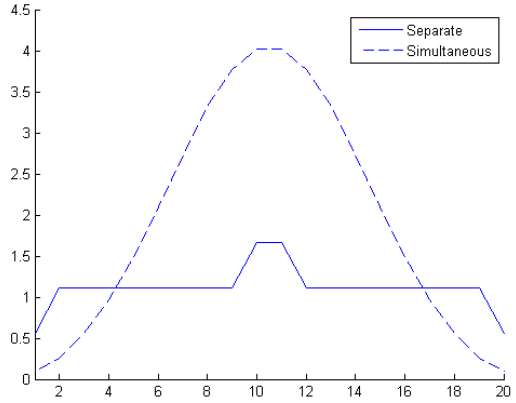

Fig. 3. Node Weight Profiles for Chain Graph

B. Eight Node Two Cluster Graph

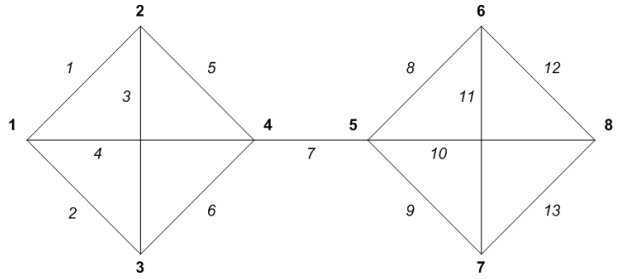

Fig. 4. Eight Node Two Cluster Graph

The second example is a graph consisting of two complete four node clusters connected by a single edge from one of the vertices in the first cluster to one in the second as in Figure 4. The Fiedler eigenvalue for such a graph is

$$\lambda_2 = 0.3542 \tag{32}$$

and the conditioning ratio is 15.9394.

For the tests, again the lowest eigenvalue bound was set to λ_2. The edge weighting resulted in a slight improvement in conditioning ratio to 13.9300. The node weighting pattern is very regular and corresponds somewhat to the linear chain case. The conditioning ratio reached by node weighting was 12.0017.

The simultaneous optimization again resulted in an improved conditioning ratio at 3.1106. The edge weighting profile denotes clear emphases and de-emphases. The node weighting shows a similar pattern to the separate case. The profiles can be seen in Figures 5 and 6.

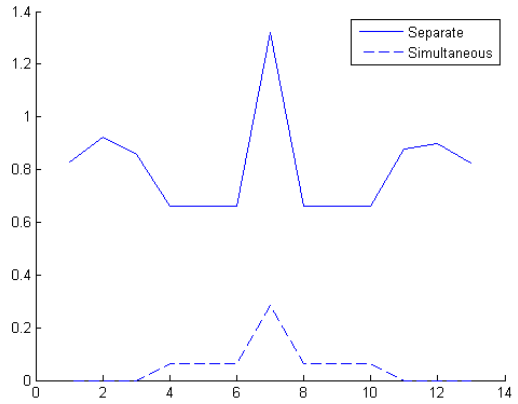

Fig. 5. Edge Weight Profiles for Eight Node Graph

C. Six Node Cycle Graph with Diameter Edge

The third example is a graph consisting of a cycle with an extra diameter edge as in figure 7. The Fiedler eigenvalue

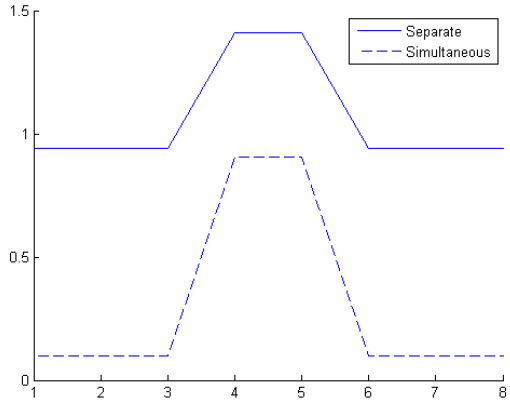

Fig. 6. Node Weight Profiles for Eight Node Graph

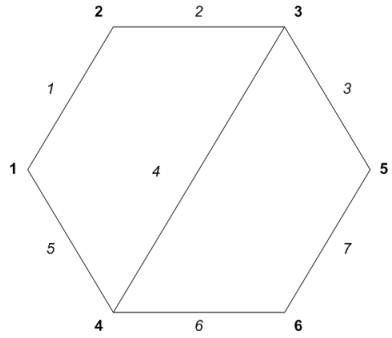

Fig. 7. Six Node Cycle Graph with Diameter Edge

for such a graph is

$$\lambda_2 = 1 \tag{33}$$

and the conditioning ratio is 5.

Problems (26) and (28) were considered with the largest eigenvalue bound set to 1. Edge weighting results in a conditioning ratio of 3.8285. Node weighting results in a conditioning ratio of 3.3333.

The simultaneous optimization results in a better conditioning ratio at 2.4228. Comparing the independent and simultaneous solutions, we note that the edge weighting profiles are somewhat inverted while the node weighting profiles are similar to the separate experiments. The results are shown in Figures 8 and 9.

VI. Vehicle Platoons

The problem of one dimensional vehicle string instability is a key issue for automated traffic management systems on highways. Given a finite grouping of vehicles arranged in a line parallel to the direction of motion, we seek to design a decentralized controller where each vehicle keeps the spacing between its forward and rear neighbors fixed while maintaining a constant speed.

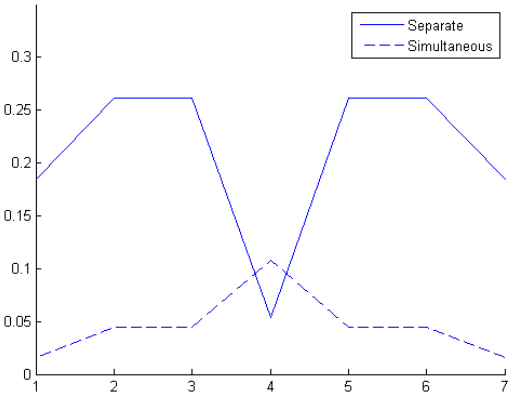

Fig. 8. Edge Weight Profiles for Cycle Graph

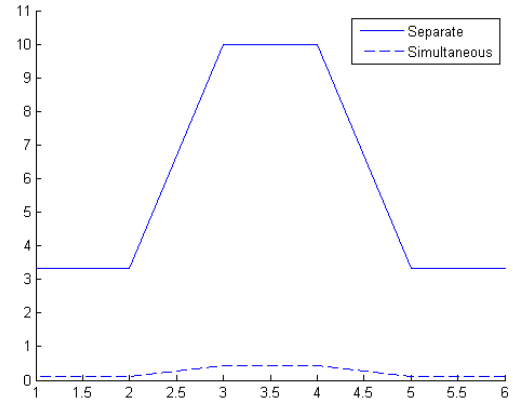

Fig. 9. Node Weight Profiles for Cycle Graph

We assume that each vehicle has identical double integrator dynamics described by the transfer function

$$H(s) = \frac{1}{s^2}. \tag{34}$$

Then the input-output relation for each vehicle is given by

$$Y_i(s) = H(s)U_i(s). \tag{35}$$

We define a common lead controller that allows each vehicle to coordinate its motion with its predecessor and follower

$$K(s) = \frac{s+1}{0.09s+1}, \tag{36}$$

and write the feedback input for the entire system as

$$U(s) \quad = \quad (L \otimes K(s))\, Y(s) \tag{37}$$

where Y is the vector of all vehicles ordered from first to last, L is the Laplacian matrix representation of the graph representing the vehicle string, and $\otimes$ denotes the familiar Kronecker product. From 35 and 37, the closed loop poles are the roots of the polynomial

$$det(I - H(s)K(s)L) = 0 \tag{38}$$

We diagonalize L and determine the roots of 38 from

$$1 - \lambda_i H(s)K(s) = 0, \;\; i = 1, \ldots n \tag{39}$$

where λ_i are the eigenvalues of L. Then the stability of the system is equivalent to the stability of the roots of 39.

We study the root locus (Figure 10) for (6) by varying λ_i, and require the closed loop system to have a settling time of less than four seconds and a damping ratio of 0.60. Based on these design choices, the smallest gain λ that satisfies the design constraints is 1.84, at which point the damping ratio is 0.66 and the natural frequency is 1.50. The largest gain that satisfies the design constraints is 7.22, at which point the damping ratio is 0.60 and the natural frequency is 8.28. This means that all eigenvalues $\lambda_i(L)$ must lie in the interval $[1.84, 7.22]$. However, the Laplacian for a string graph with identical node and edge weights does not satisfy this constraint.

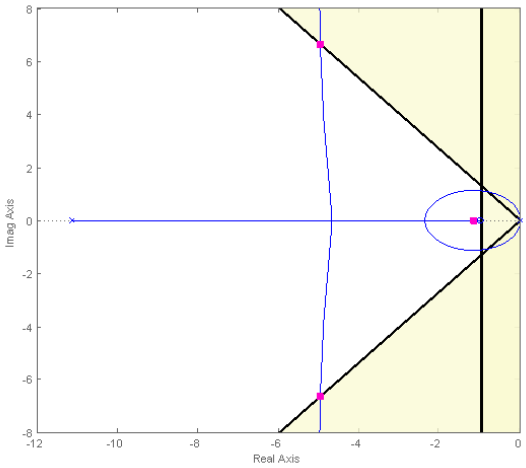

Fig. 10. Root Locus for $H(s)K(s)$ with poles set to maximum acceptable compensation

We now use semidefinite programming to find feasible edge and node weightings such that $\lambda_i(L_g) \in [1.84, 7.22]$. The semidefinite program shows that we can achieve the specified performance criteria for up to five nodes.

For a five node graph, the conditioning ratio is reduced from 9.47 to 3.56, and so the gap can be achieved. In [17], it was shown that a symmetric bidirectional controller scheme, in which each vehicle uses spacing information from its predecessor and follower, suffers from inscalability problems. We have shown how a judicious choice of node and edge weighting achieved via semidefinite programming can be used to mitigate scalability issues.

VII. Conclusion

The spectrum of the Laplacian is an invaluable indicator for dynamical properties of a networked system. In the paper,

a novel approach to bound the gap between the smallest and largest eigenvalues of a graph was discussed. By use of convex optimization, it was possible to simultaneously adjust the edge and node weightings for a graph and subject the graph to both upper and lower eigenvalue constraints. The analysis tools outlined are useful for a variety of stability and control problems in decentralized systems. We have a number of possible directions to take for extensions to this work. For instance, numerical simulation shows that our convex relaxation is nearly optimal for most graphs. We will conduct a rigorous study to quantitatively characterize any optimality gaps. We are also investigating other physical and biological application systems that can be optimized through this framework, as well as ways to incorporate associated additional design constraints. From an algorithms standpoint, we are examining formulations of the dual leading to faster solution methods for large networks.

REFERENCES

[1] F. Chung, "Spectral graph theory," *American Mathematical Society*, 1997.

[2] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[3] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[4] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[5] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[6] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[7] M. Arcak, "Passivity as a design tool for group coordination," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1380–1390, 2007.

[8] H. Bai and M. Arcak, "Instability mechanisms in cooperative control," *IEEE transactions on automatic control*, vol. 55, no. 1, pp. 258–263, 2010.

[9] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[10] A. Ghosh, S. Boyd, and A. Saberi, "Minimizing effective resistance of a graph," *SIAM review*, vol. 50, no. 1, p. 37, 2008.

[11] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM review*, vol. 46, no. 4, pp. 667–689, 2004.

[12] J. Sun, S. Boyd, L. Xiao, and P. Diaconis, "The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem," *SIAM review*, vol. 48, no. 4, pp. 681–699, 2006.

[13] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, p. 117, 2006.

[14] R. Horn and C. Johnson, *Matrix analysis*. Cambridge Univ Pr, 1990.

[15] F. Gantmacher, *The theory of matrices*. Chelsea Pub Co, 2000.

[16] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21," http://cvxr.com/cvx, Sep. 2010.

[17] P. Seiler, A. Pant, and K. Hedrick, "Disturbance propagation in vehicle strings," *IEEE Transactions on Automatic Control*, vol. 49, no. 10, pp. 1835–1842, 2004.