Proceedings of the 45th IEEE Conference on Decision & Control
Manchester Grand Hyatt Hotel
San Diego, CA, USA, December 13-15, 2006

Growing Well-connected Graphs

Arpita Ghosh Stephen Boyd
Information Systems Laboratory
Stanford University
arpitag, boyd@stanford.edu

Abstract— **The algebraic connectivity of a graph is the second smallest eigenvalue of the graph Laplacian, and is a measure of how well-connected the graph is. We study the problem of adding edges (from a set of candidate edges) to a graph so as to maximize its algebraic connectivity. This is a difficult combinatorial optimization, so we seek a heuristic for approximately solving the problem. The standard convex relaxation of the problem can be expressed as a semidefinite program (SDP); for modest sized problems, this yields a cheaply computable upper bound on the optimal value, as well as a heuristic for choosing the edges to be added. We describe a new greedy heuristic for the problem. The heuristic is based on the Fiedler vector, and therefore can be applied to very large graphs.**

I. Introduction

Let $G = (V, E)$ be an undirected graph with n nodes and m edges. For an edge l connecting nodes i and j, we define the edge vector $a_l \in \mathbf{R}^n$ as $a_{l_i} = 1$, $a_{l_j} = -1$, and all other entries 0. The incidence matrix $A \in \mathbf{R}^{n\times m}$ of the graph G is the matrix with lth column a_l. The Laplacian L of G is the $n \times n$ matrix

$$L = AA^T = \sum_{l=1}^{m} a_l a_l^T, \qquad (1)$$

i.e., the diagonal entry L_{ii} is the degree of node i, and $L_{ij} = -1$ if $(i, j) \in E$, and is 0 otherwise. Clearly, $L \succeq 0$ (*i.e.*, it is positive semidefinite), and $L\mathbf{1} = 0$, where $\mathbf{1}$ is the vector of all ones. Let $0 = \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ be the eigenvalues of L. The second smallest eigenvalue $\lambda_2(L)$ is called the *algebraic connectivity* of the graph G, and the corresponding normalized eigenvector is called the *Fiedler vector* [Fie73].

There are many reasons the algebraic connectivity is considered to be a measure of how well-connected a graph is. For one, $\lambda_2(L)$ is monotone increasing in the edge set: if $G_1 = (V, E_1)$ and $G = (V, E_2)$ are such that $E_1 \subseteq E_2$, then $\lambda_2(L_1) \le \lambda_2(L_2)$ [Fie73]. That is, the more connected graph (on the same vertex set) has the greater algebraic connectivity.

The algebraic connectivity has a direct connection to the number of connected components: $\lambda_2(L) > 0$ if and only if G is connected [Fie73]; in fact, the mutiplicity of the 0 eigenvalue is exactly equal to the number of connected components in G.

The algebraic connectivity is also related to the sparsity of cuts in the graph. Let X be a proper subset of V, and let E_c be the set of edges connecting X and its complement X^c. Then [FKP03]

$$\lambda_2(L) \le \min_{X\subseteq V} \frac{n|E_c|}{|X||X^c|}.$$

That is, a graph with large algebraic connectivity cannot have very sparse cuts; conversely, λ_2 is small for a graph with sparse cuts. This is related to a heuristic from graph partitioning, whereby an approximate bisection of a graph is computed using the Fiedler vector. Roughly speaking, a graph with small algebraic connectivity is easier to bisect than one with large algebraic connectivity. We discuss this in greater detail in §III-C. There is also a relation between $\lambda_2(L)$ and linear embeddings of graphs: a graph with large $\lambda_2(L)$ cannot be embedded very well into $\mathbf{R}$, and vice-versa. This connection is discussed in greater detail in §III-D.

The mixing rate of a continuous time Markov chain with unit rates on the edges is given by $\lambda_2(L)$; so this Markov chain converges to its stationary distribution faster for a graph with larger algebraic connectivity. The algebraic connectivity has also emerged as an important parameter in many systems problems defined over networks, notably as a measure of stability and robustness of the networked dynamic systems; for a number of references, see [KM06].

The paper is organized as follows. In the remainder of this section, we state the problem and discuss related work. We derive the convex relaxation in §II. In §III, we derive a heuristic based on the Fiedler vector and discuss its properties. We compare the performance of this heuristic in §IV. Extensions are discussed in §V.

A. Problem statement

In this paper, we study the following problem. Given a base graph $G_{\text{base}} = (V, E_{\text{base}})$, a set of m_c candidate edges E_{cand} on V, and a number k, $0 \le k \le m_c$, choose

k edges from E_{cand} that lead to the greatest increase in algebraic connectivity when added to G_{base}. (We assume that $E_{\text{base}} \cap E_{\text{cand}} = \emptyset$.) That is, we want to solve the problem

$$\begin{array}{ll} \text{maximize} & \lambda_2(L(E_{\text{base}} \cup E)) \\ \text{subject to} & |E| = k, \\ & E \subseteq E_{\text{cand}}, \end{array} \tag{2}$$

where the optimization variable is the subset E of candidate edges. (Note that if we replace the constraint $|E| = k$ by $|E| \leq k$, there will be an optimal solution with $|E| = k$ since $\lambda_2(L)$ is monotone in the edge set.)

This can be formulated as a Boolean problem as follows. Every subset of E_{cand} can be encoded using a Boolean vector $x \in \{0,1\}^{m_c}$, where $x_l = 1$ if edge l belongs to the subset, and $x_l = 0$ otherwise. Let L_{base} denote the Laplacian matrix corresponding to G_{base}. Then (2) can be written as

$$\begin{array}{ll} \text{maximize} & \lambda_2(L_{\text{base}} + \sum_{l=1}^{m_c} x_l a_l a_l^T) \\ \text{subject to} & \mathbf{1}^T x = k, \\ & x \in \{0,1\}^{m_c}, \end{array} \tag{3}$$

with variable x.

This problem is combinatorial, and can, of course, be solved exactly by exhaustive search over a set of size $\binom{m_c}{k}$, *i.e.*, by computing λ_2 for $\binom{m_c}{k}$ Laplacian matrices. However, this is not practical for large m_c and k; here we are interested in efficient heuristics for this problem.

B. Related Work

The algebraic connectivity of a graph has been studied extensively since the seminal work of Fiedler [Fie73]. However, there have only been a few studies in the literature dealing with the problem of choosing a graph from a set of graphs to maximize the algebraic connectivity. In [FK98], the authors use graph-theoretic methods to extremize the algebraic connectivity over trees of fixed diameter. In [KM06], the authors study the problem of choosing the vertex configuration subject to a proximity constraint that maximizes the algebraic connectivity. In [DM05], the authors use the relation between algebraic connectivity and K-connectivity to formulate the problem of finding power-efficient topologies subject to constraints on K-connectivity as $0/1$ semidefinite programs. A related article of interest is [Guo06], where the author studies the effect of adding and grafting edges on the spectral radius of the Laplacian.

A related problem is that of optimizing weights on the edges of a graph to maximize the algebraic connectivity of the weighted graph (here the variables are the weights on the edges, and not the graph topology). Fiedler defines this optimal value as the absolute algebraic connectivity of the graph [Fie93]. For related work on the algebraic connectivity, see [SBXD] and [GHW05].

II. Convex relaxation

Here we derive the convex relaxation of (3). For $x \in [0,1]^{m_c}$, define

$$L(x) = L_{\text{base}} + \sum_{l=1}^{m_c} x_l a_l a_l^T. \tag{4}$$

Replacing the Boolean constraint $x \in \{0,1\}^{m_c}$ by the linear constraint $x \in [0,1]^{m_c}$, we obtain the following relaxation of (3):

$$\begin{array}{ll} \text{maximize} & \lambda_2(L(x)) \\ \text{subject to} & \mathbf{1}^T x = k, \\ & 0 \leq x \leq 1. \end{array} \tag{5}$$

Since this problem has a larger feasible set than (3), its optimal value is an upper bound on the optimal value of (3).

The problem (5) is a convex optimization problem: the constraints are linear functions of the variable x, and the objective is a concave function of x. To see this, note that the matrix $L(x)$ is positive semidefinite on $[0,1]^{m_c}$, and has smallest eigenvalue 0 with corresponding eigenvector $\mathbf{1}$. So its second smallest eigenvalue is ([HJ85, §4.2])

$$\lambda_2(L(x)) = \inf\{y^T L(x) y \mid \|y\| = 1, \mathbf{1}^T y = 0\}.$$

So $\lambda_2(L(x))$ is the pointwise infimum of a family of linear functions of x, and is therefore a concave function of x ([BV04, §3.2.3]).

The convex relaxation (5) can be formulated as the following semidefinite program (SDP):

$$\begin{array}{ll} \text{maximize} & s \\ \text{subject to} & s(I - \mathbf{1}\mathbf{1}^T/n) \preceq L(x), \\ & \mathbf{1}^T x = k, \\ & 0 \leq x \leq 1. \end{array} \tag{6}$$

A standard SDP solver can be used to solve (6) for moderate problem sizes, where m_c is up to a 1000 edges or so. For larger problems, a subgradient method can be used, but this may converge slowly to the optimal solution to the relaxation. The computational requirements for solving (6) motivate the local heuristic in §III.

Solving (5) immediately gives an upper bound on the optimal value of (3); if in addition the solution to (5) is Boolean, it is also optimal for (3). In general, this need not happen, and we use a heuristic to obtain a Boolean vector from the optimal solution to (5). One simple heuristic is to set the largest y_l to 1, and the remaining entries to 0. More sophisticated heuristics such as randomized rounding [Vaz03] may be used as well.

We note that the solution to (5) is a *concave* function of k, the number of edges added. (The optimal value of a concave maximization problem with constraints $\mathcal{Q}(x) \leq \theta$ is a concave function of θ [Lue69]). Roughly, this means that to the extent to which the upper bound reflects the actual optimal value of (3), the first few edges added lead to a much greater increase in algebraic connectivity than those added later on. That is, a large increase in algebraic connectivity can be obtained by only adding a few well-chosen edges.

III. A GREEDY PERTURBATION HEURISTIC

Here we describe a greedy local heuristic for solving (3) based on the Fiedler vector. We add the k edges one at a time, each time choosing the edge $l \sim (i,j)$ which has the largest value of $(v_i - v_j)^2$, where v is a Fiedler vector of the current Laplacian.

Greedy Heuristic:

- Starting from G_{base}, add k edges, one at a time:
  1. Find v, a unit eigenvector corresponding to $\lambda_2(L)$, where L is the current Laplacian.
  2. From the remaining candidate edges, add an edge (i,j) with largest $(v_i - v_j)^2$.

The motivation for this heuristic is as follows.

If v is an eigenvector with unit norm corresponding to λ_2, then vv^T is a *supergradient* of λ_2 at L, *i.e.*, for any symmetric matrix Y

$$\lambda_2(L+Y) \leq \lambda_2(L) + \mathbf{Tr}(Yvv^T).$$

If λ_2 is isolated (*i.e.*, $\lambda_1 < \lambda_2 < \lambda_3$), then $\lambda_2(L)$ is an analytic function of L, and therefore of x. In this case the supergradient is the gradient, *i.e.*,

$$\frac{\partial}{\partial x_l}\lambda_2(L) = v^T \frac{\partial L}{\partial x_l} v, \tag{7}$$

where v is the unique normalized eigenvector (up to a sign flip) corresponding to λ_2. (This result also follows from perturbation theory for symmetric matrices.) Since

$$\frac{\partial L}{\partial x_l} = a_l a_l^T,$$

the partial derivative of $\lambda_2(L)$ with respect to x_l is $(v_i - v_j)^2$, where $l \sim (i,j)$. In other words, when $\lambda_2(L)$ is isolated, $(v_i - v_j)^2$ gives the first order approximation of the increase in $\lambda_2(L)$, if edge l is added to the graph. Therefore step (2) in our algorithm corresponds to adding an edge, from among the remaining candidates, that gives the largest predicted increase in $\lambda_2(L)$, according to a first order approximation.

When λ_2 is not isolated, adding a single edge cannot increase λ_2. It follows directly from the results in [BNS78] that if $\lambda_2 > 0$ has multiplicity r, exactly r edges need to be added before λ_2 increases from its current value.

A. Computing the perturbation heuristic

The perturbation heuristic requires only an eigenvector computation, as opposed to the solution of an SDP for the convex relaxation. Computing all the eigenvectors of an $n \times n$ dense matrix costs approximately $4/3n^3$ arithmetic operations. Lanczos algorithms can be used for very efficiently computing the top and bottom few eigenvalues and approximate eigenvectors of large sparse matrices [Dem97]. Given an $n \times n$ sparse symmetric matrix P, the Lanczos algorithm computes a $h \times h$ symmetric tridiagonal matrix T, whose eigenvalues are good approximations of h eigenvalues of P, and whose eigenvectors can be used to compute approximate eigenvectors of P. Conveniently, the first eigenvalues to converge are the largest and smallest few eigenvalues of P. Also, since we are only interested in the relative values of $(v_i - v_j)^2$ for candidate edges (i,j), the Fiedler vector does not need to be computed to high accuracy. Thus, the Lanczos algorithm is a good choice for our perturbation heuristic.

The Fiedler vector can also be computed efficiently for very large graphs using algebraic multigrid techniques as described in [KCH02]. The algorithm described in [KCH02] proceeds by expressing the original high-dimensional problem in lower and lower dimensions using a *coarsening* process, and solves the problem exactly at the coarsest level. A *refinement* process is then used to project the solution back to higher dimensions, appropriately updating the solution at each scale, until the original problem has been solved.

B. Properties

We discuss two properties of the perturbation heuristic in this section.

1) Connecting disconnected graphs: Suppose a graph is disconnected. Then the local heuristic applied to this graph always chooses an edge that connects two connected components.

To see this, suppose G has r connected components. Then 0 is an eigenvalue with multiplicity r. The set of vectors $v^1, \ldots, v^r$, where $v_j^q = 1/\sqrt{n_q}$ if node j belongs to the connected component q with size n_q, and 0 if it does not, is an orthonormal set of eigenvectors for the eigenvalue 0. Any linear combination of these eigenvectors (except multiples of $\mathbf{1}$) has $(v_i - v_j)^2 = 0$ for i and j belonging to the same component, and there is some pair of nodes i and j belonging to different connected components for which $(v_i - v_j)^2 > 0$. So the local heuristic will always pick an edge across components. Specifically, λ_2 changes from 0 with the $r-1$th edge added by the local heuristic, since each edge chosen by the local heuristic bridges two connected components.

2) Lower and upper bounds: Here we derive lower and upper bounds on the algebraic connectivity of a graph obtained by adding a single edge to a connected graph.

Let L be the Laplacian of the current graph. We assume that $\lambda_2(L)$ is isolated, with normalized eigenvector v. Suppose we add an edge $l \sim (i,j)$ with edge vector a_l to this graph. Then, we have the following bounds for $\lambda_2(L + a_l a_l^T)$:

$$\lambda_2(L + a_l a_l^T) \geq \lambda_2 + \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}, \qquad (8)$$

and

$$\lambda_2(L + a_l a_l^T) \leq \lambda_2 + \frac{(v_i - v_j)^2}{1 + (2 - (v_i - v_j)^2)/(\lambda_n - \lambda_2)}. \qquad (9)$$

Note that lower and upper bound are both increasing functions of $(v_i - v_j)^2$. Therefore, the perturbation heuristic can be interpreted as adding the edge to the current graph that maximizes both the lower bound (8) and the upper bound (9) on the algebraic connectivity of the resulting graph.

A weaker upper bound can be obtained immediately as follows: Since vv^T is a supergradient of $\lambda_2(L)$,

$$\begin{aligned} \lambda_2(L + a_l a_l^T) &\leq \lambda_2(L) + v^T(L + a_l a_l^T)v \\ &= \lambda_2(L) + (v_i - v_j)^2. \end{aligned} \qquad (10)$$

Lower bound:
The lower bound is established as follows. Let $L = Q\Lambda Q^T$ be the eigenvalue decomposition of L, and suppose that all entries in Λ are distinct (the same bound holds if eigenvalues other than λ_2 are repeated). The matrices L and $L + a_l a_l^T$ both have eigenvalue 0 with corresponding eigenvector $\mathbf{1}$. The remaining $n-1$ eigenvalues of $L + a_l a_l^T$ are the $n-1$ roots of the secular equation ([BNS78], [Gol73])

$$\sum_{i=2}^{n} \frac{u_i^2}{d - \lambda_i} = 1, \qquad (11)$$

where $u = Q^T a_l$ (note that $u_2 = v_i - v_j$, and $u_1 = 0$).

Denote by $d_2, \ldots, d_n$ the remaining ordered eigenvalues of $L + a_l a_l^T$. By eigenvalue interlacing,

$$\lambda_i < d_i < \lambda_{i+1}, \quad i = 2, \ldots, n-1.$$

Now consider d in the interval (λ_2, λ_3). For d in this interval, if

$$\frac{u_2^2}{d - \lambda_2} \geq 1 + \sum_{i=3}^{n} \frac{u_i^2}{\lambda_i - d},$$

then $d \leq d_2$. Since $\|u\| = 2$, $\sum_{i=3}^n u_i^2 \leq 2$, so if

$$\frac{u_2^2}{d - \lambda_2} \geq 1 + \frac{2}{\lambda_3 - d},$$

then $d \leq d_2$. Setting $d - \lambda_2 = \epsilon$, and $\lambda_3 - \lambda_2 = \delta$, we want to find $\epsilon > 0$ such that

$$\frac{u_2^2}{\epsilon} \geq 1 + \frac{2}{\delta - \epsilon}. \qquad (12)$$

It can be verified that setting

$$\epsilon = \frac{u_2^2}{2(1/\delta + 1/4 + \sqrt{1/4 + 4/\delta^2})}$$

satisfies (12). Since $\sqrt{1/4 + 4/\delta^2} \leq 1/2 + 2/\delta$, we obtain the lower bound in (8).

Upper bound:
A sharper upper bound than (10) can be obtained using the secular equation. From (11), the algebraic connectivity of $L + a_l a_l^T$ is the number $d_2 \in (\lambda_2, \lambda_3)$ satisfying

$$d = \lambda_2 + \frac{u_2^2}{1 + \sum_{i=3}^n u_i^2/(\lambda_i - d)}.$$

The LHS is an increasing function of d and the RHS is decreasing in d. So using any $d \leq d_2$ in the RHS gives an upper bound on d_2. Using $d = \lambda_2 < d_2$, we obtain the following upper bound:

$$\begin{aligned} d_2 &\leq \lambda_2 + \frac{u_2^2}{1 + \sum_{i=3}^n u_i^2/(\lambda_i - \lambda_2)} \\ &\leq \lambda_2 + \frac{u_2^2}{1 + (2 - u_2^2)/(\lambda_n - \lambda_2)}. \end{aligned}$$

This gives the upper bound in (9).

C. Relation to spectral graph partitioning

A well-known heuristic for spectral graph partitioning is based on the Fiedler vector [PSL90], [Chu97]. The nodes are divided into sets N_1 and N_2 based on the Fiedler vector as follows:

$$N_1 = \{i \mid v_i < 0\}, \qquad N_2 = \{i \mid v_i \geq 0\}.$$

(A more general version of this heuristic partitions the nodes according to whether v_i is greater or less than α, where α sets the relative sizes of the partition.)

A node with a large positive or large negative value of v_i belongs 'strongly' in N_1 or N_2, whereas nodes with v_i close to 0 do not belong strongly to either side of the partition. The perturbation heuristic chooses nodes that most strongly belong to different components, *i.e.*, with large $(v_i - v_j)^2$, and adds an edge between them.

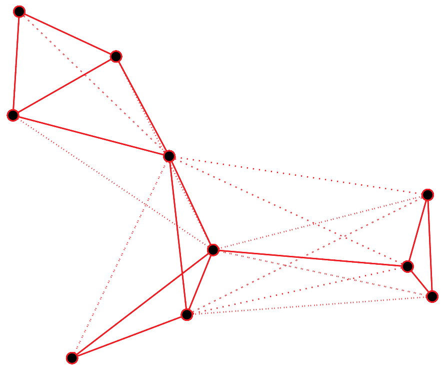

Fig. 1. A randomly generated graph with 10 nodes, 14 edges, and $m_c = 11$.

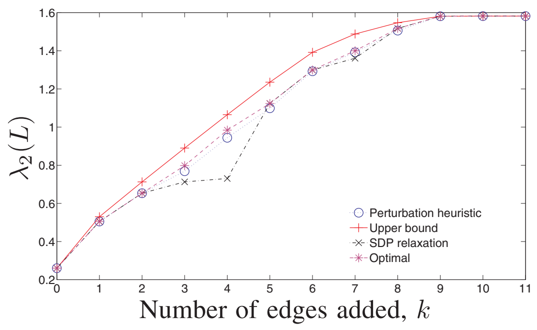

Fig. 2. A comparison of the heuristics described with the optimal solution and upper bound for the graph in Figure 1.

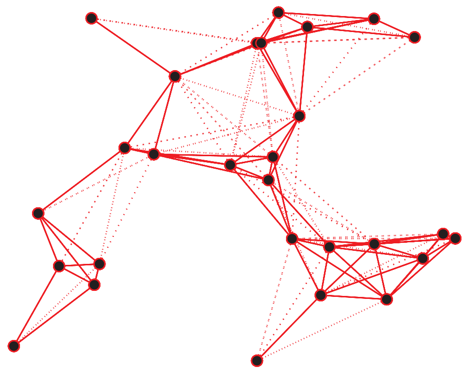

Fig. 3. A randomly generated graph with 28 nodes, 68 edges, and $m_c = 42$.

D. Relation to linear embedding

Another interpretation of the perturbation heuristic is via linear embedding. The problem of embedding a graph $G = (V, E)$ into the real line is the following: assign node coordinates $x_1, \ldots, x_n$, with 0 mean and unit variance, so as to minimize the sum of squares of distances between adjacent nodes. That is, the real embedding problem is

$$\begin{array}{ll} \text{minimize} & \sum_{(i,j)\in E}(x_i - x_j)^2 \\ \text{subject to} & \mathbf{1}^T x = 0, \\ & x^T x = 1. \end{array} \qquad (13)$$

The unit variance constraint is imposed since otherwise the problem is trivial with optimal solution $x_i = 0$, and optimal value 0.

The objective in (13) is simply $x^T Lx$, and so the optimal value is $\lambda_2(L)$, with optimal solution $x = v$, where v is the Fiedler vector. The perturbation heuristic therefore adds an edge between the nodes that are farthest from each other in the linear embedding.

IV. Examples

Figure 1 shows a randomly generated graph with 10 nodes and 14 edges, and $m_c = 11$. The edges in E_{base} are shown as solid lines, and the edges in E_{cand} are dotted. For this small example, we compute the optimal solution to (3) by exhaustive search. We compare this with the two heuristics, and the upper bound computed from the SDP relaxation in Figure 2. We see that the results from the local heuristic are very close to the actual optimal value.

Figure 3 shows a larger example with 28 nodes and 68 edges, and $m_c = 42$. The performance of the heuristics is compared with the upper bound in Figure 4. Again, the results from the local heuristic are very close to the upper bound from the convex relaxation.

Finally, we show an example on a randomly generated graph with 1000 nodes and 5517 edges, and $m_c = 2341$ in Figure 5. For this graph, we compare addition using the perturbation heuristic with random addition in Figure 6. As expected, random addition performs very badly compared to the perturbation heuristic. However, the point to be noted here is the following: after adding 150 edges using the perturbation heuristic, the algebraic connectivity increases to 0.041, while the algebraic connectivity after adding *all* 2341 edges is only 0.051. This suggests that a large increase in algebraic connectivity can be obtained by adding a few edges carefully (this observation correlates well with the concavity of the upper bound).

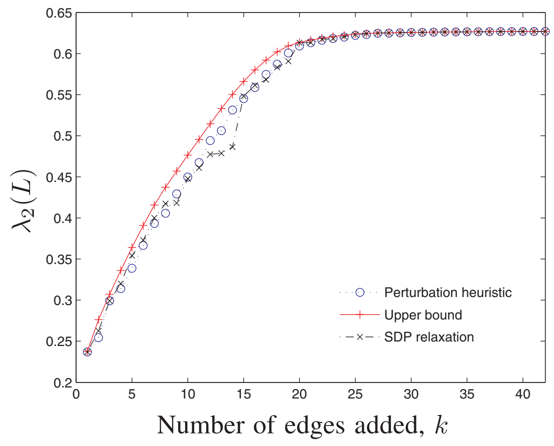

Fig. 4. A comparison of the heuristics described and the upper bound from the SDP relaxation for the graph in Figure 3.

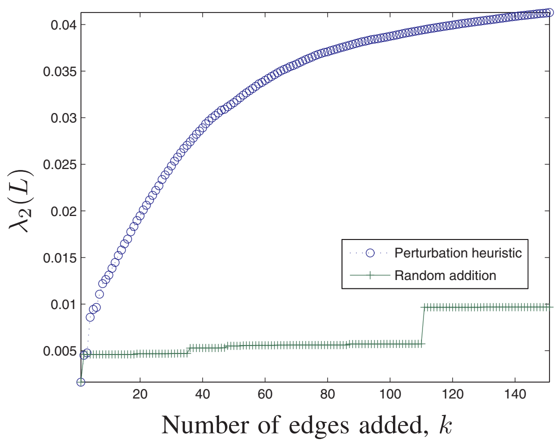

Fig. 6. A comparison of the perturbation heuristic with random addition for the graph in Figure 5.

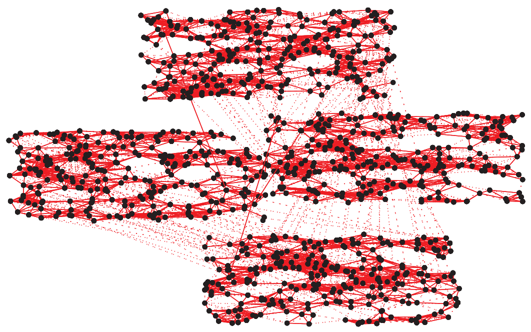

Fig. 5. A randomly generated graph with 1000 nodes, 5517 edges, and $m_c = 2341$.

V. Extensions

A simple extension of the problem is to associate a *cost* c_l with each edge l. In this case, there is a constraint on the total cost of the edges added, *i.e.*, the problem is

$$\begin{array}{ll} \text{maximize} & \lambda_2(L + \sum_{l=1}^{m_c} x_l a_l a_l^T) \\ \text{subject to} & c^T x \le k, \\ & x \in \{0,1\}^{m_c}. \end{array} \quad (14)$$

Such a problem might arise, for example, in wireless networks, where the cost of a link is the power required to transmit across that link; the cost could also be a function of some geometric distance between two nodes.

The convex relaxation is obtained exactly as before. The perturbation heuristic is modified as follows: changing variables to $y_l = c_l x_l$, the problem is to choose k of the y_l to maximize $\lambda_2(L + \sum_{l=1}^{m_c}(1/c_l) y_l a_l a_l^T)$. If λ_2 is isolated, then

$$\frac{\partial \lambda_2}{\partial y_l} = \frac{1}{c_l}(a_l^T v)^2.$$

In this case, at each step we add $l \sim (i,j)$ from among the candidates with the largest $(v_i - v_j)^2/c_l$, until no edge can be added without the total cost exceeding k.

We also remark that the same idea works for *removing* edges from a given graph: remove the k edges one at a time by computing a Fiedler vector for the current Laplacian, and choose the edge with the smallest $(v_i - v_j)^2$. This heuristic also performs well, like the heuristic for adding edges. Finally, we remark that we can construct an algorithm that adds and removes edges in order to maximize the algebraic connectivity of the graph.

References

[BNS78] J. Bunch, C. Nielsen, and D. Sorensen. Rank one modification of the symmetric eigenproblem. *Numerical Mathematics*, 31:31–48, 1978.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. Available at `www.stanford.edu/~boyd/cvxbook`.

[Chu97] F. Chung. *Spectral Graph Theory*. AMS, 1997.

[Dem97] J. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[DM05] A. Das and M. Mesbahi. k-node connected power efficient topologies in wireless networks: A semidefinite programming approach. *UWEE Technical Report*, (UWEETR-2005-0004), February 2005.

[Fie73] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematics Journal*, 23:298–305, 1973.

[Fie93] M. Fiedler. Some minimax problems for graphs. *Discrete Mathematics*, 121:65–74, 1993.

[FK98] S. Fallat and S. Kirkland. Extremizing algebraic connectivity subject to graph theoretic constraints. *The Electronic Journal of Linear Algebra*, 3:48–74, 1998.

[FKP03] S. Fallat, S. Kirkland, and S. Pati. On graphs with algebraic connectivity equal to minimum edge density. *Linear Algebra Appl.*, 373:31–50, 2003.

[GHW05] F. Göring, C. Helmberg, and M. Wappler. Embedded in the shadow of the separator. Preprint 2005-12, Fakultät für Mathematik, Technische Universität Chemnitz, Chemnitz, Germany, September 2005.

[Gol73] G. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15:318–334, 1973.

[Guo06] J. Guo. The effect on the Laplacian spectral radius of a graph by adding or grafting edges. *Linear Algebra Appl.*, 413:59–71, 2006.

[HJ85] R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[KCH02] Y. Koren, L. Carmel, and D. Harel. Ace: A fast multiscale eigenvectors computation for drawing huge graphs. 2002. Technical Report MCS01-17, The Weizmann Institute of Science.

[KM06] Y. Kim and M. Mesbahi. On maximizing the second-smallest eigenvalue of a state dependent graph Laplacian. *IEEE Trans. Automatic Control*, 51(1):116–120, January 2006.

[Lue69] D. Luenberger. *Optimization by Vector Space Methods*. John Wiley and Sons, 1969.

[PSL90] A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, (11):430–452, 1990.

[SBXD] 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. To appear, *SIAM Review, Problems and Techniques Section*. Available online at `www.stanford.edu/~boyd/fmmp.html`.

[Vaz03] V. Vazirani. *Approximation Algorithms*. Springer, New York, 2003.