Bisection algorithm of increasing algebraic connectivity by adding an edge

Yoonsoo Kim

***Abstract*— For a given graph (or network) G, consider another graph G' by adding an edge e to G. We propose a computationally efficient algorithm of finding e such that the second smallest eigenvalue (algebraic connectivity, $\lambda_2(G')$) of G' is maximized. Theoretically, the proposed algorithm runs in $O(4mn\log(d/\epsilon))$, where n is the number of nodes in G, m is the number of disconnected edges in G, d is the difference between $\lambda_3(G)$ and $\lambda_2(G)$, and $\epsilon > 0$ is a sufficiently small constant. However, extensive simulations show that the *practical* computational complexity of the proposed algorithm, $O(5.7mn)$, is nearly comparable to that of a simple greedy-type heuristic, $O(2mn)$. This algorithm can also be easily modified for finding e which affects $\lambda_2(G)$ the least.**

Key words: algebraic connectivity; Laplacian matrix.

I. Introduction

Suppose a graph or network G is given which consists of the set of nodes, $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, and the set of edges, $\mathcal{E} = \{e_{ab} \,|\, v_a, v_b \in \mathcal{V}\}$. Consider then another graph G' in which the set of nodes is the same as $\mathcal{V}$ but the set of edges is the union of $\mathcal{E}$ and an edge $e_{ij} \in \mathcal{E}^c$ (i.e. $e_{ij} \notin \mathcal{E}$), where $v_i, v_j \in \mathcal{V}$. In this note, we are interested in finding e_{ij} such that the second smallest eigenvalue (algebraic connectivity, $\lambda_2(G')$) of the graph Laplacian $L_{G'}$ is maximized. As will become clear, similar problems, e.g. finding e_{ij} such that $\lambda_2(G')$ is minimized, can be easily handled by modifying the method to be discussed in this note.

The algebraic connectivity $\lambda_2(G)$, originally defined in [4], has been identified as an important parameter in many systems problems defined over networks [5], [13], [14], [15], [16] (see [7] for a good introduction to the algebraic connectivity). It is interesting to note that $\lambda_2(G)$ is also used for biological applications, e.g. [1]. In this note, we are particularly interested in using $\lambda_2(G)$ for optimal network design. This interest is motivated by the observation that $\lambda_2(G)$ is a measure of stability and robustness of the networked dynamic system [5], [13]. A few works can be found in the literature along this line. In [13], the authors use $\lambda_2(G)$ for finding the best node configuration with the largest $\lambda_2(G)$ when the nodes are subject to proximity constraints. The proximity constraints are then linearized to solve a series of semi-definite programs to find a locally optimal configuration. The same problem is approached via a sphere-packing idea in [12].

In [6], a similar problem of finding the best network but subject to a constraint on the number of edges in the network is considered. The authors propose a greedy perturbation heuristic to find the network with the largest $\lambda_2(G)$ by adding edges one by one until the constraint is satisfied. This heuristic turns out to be useful for network synchronization in [10]. In [10], the authors consider rewiring a given network by adding or deleting an edge to increase $\lambda_2(G)$ while keeping the same number of edges in the network. The same network design problem subject to the constraint on the number of edges is also considered in [11] for relay deployment in wireless sensor networks. In [11], the authors propose a semi-definite programming relaxation technique to find the best locations for a given set of relays. There are few other related works in the literature which address similar problems, such as [3], [9], but it seems that only few ideas are around to deal with the optimal network design problems of involving $\lambda_2(G)$.

As stated in the beginning, we are interested in finding the best edge which can increase the current $\lambda_2(G)$ the most. Clearly, the greedy perturbation heuristic proposed in [6] may do a *proper* job in this regard with small computational power. In this note, we however aim to propose an algorithm doing the *perfect* job at the expense of a little bit more computational power. To this end, in the following sections, we introduce a computationally efficient bisection algorithm utilizing the secular equation [2] and compare its performance to the greedy perturbation heuristic theoretically as well as numerically.

II. Methods

A. A motivating example and two existing approaches

To begin with, consider a graph G in Figure 1, where

$$\mathcal{V} = \{v_1, v_2, \ldots, v_5\} \quad \text{and} \quad \mathcal{E} = \{e_{12}, e_{13}, e_{14}, e_{15}, e_{35}, e_{45}\}.$$

The Laplacian matrix of the graph reads as follows:

$$L_G = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & -1 \\ -1 & 0 & 0 & 2 & -1 \\ -1 & 0 & -1 & -1 & 3 \end{bmatrix}.$$

Each diagonal entry of L_G represents the number of adjacent nodes (neighbours) of the corresponding node, while each off-diagonal entry represents the connectivity between each pair of nodes (-1 if connected; 0 otherwise). For this example, one can easily check that the second smallest

This work is supported by the Young Researcher Fund (Subcommittee B fund) from the University of Stellenbosch in South Africa. The author thanks to Mehran Mesbahi and Frank Janse van Vuuren for their thoughtful comments on this manuscript.

The author is with the Department of Mechanical and Mechatronic Engineering, University of Stellenbosch, Matieland 7602, South Africa. Email: *ykim@sun.ac.za*.

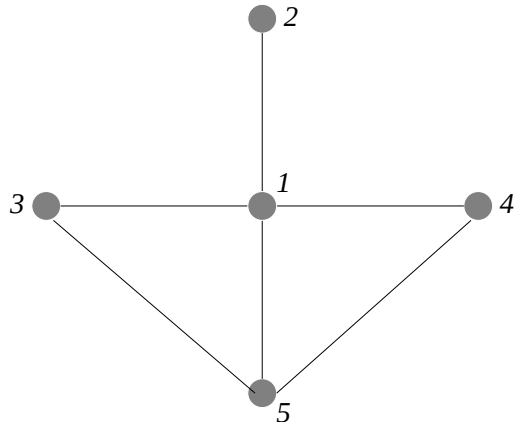


Fig. 1. A graph on 5 vertices

eigenvalue of L_G, $\lambda_2(G)$, and the third smallest eigenvalue of L_G, $\lambda_3(G)$, are 1 and 2, respectively.

We now consider choosing an edge $e \in \mathcal{E}^c = \{e_{23}, e_{24}, e_{25}, e_{34}\}$ which increases $\lambda_2(G)$ the most. It is well-known that adding an edge always yields $\lambda_2(G) \leq \lambda_2(G') \leq \lambda_3(G)$, where G' is the graph obtained by adding the edge to G, and $\lambda_2(G') = \lambda_2(G)$ if $\lambda_2(G) = \lambda_3(G)$ (see [2] for more details). Due to these relationships between $\lambda_2(G)$ and $\lambda_2(G')$, our focus in this note is on the graphs G with $\lambda_2(G) \neq \lambda_3(G)$ which guarantee $\lambda_2(G) < \lambda_2(G')$.

The easiest approach to finding e is to go over all the possible edges for each of which the corresponding $\lambda_2(G')$ is computed. Using this way, one can see that $e = e_{25}$ yields the largest $\lambda_2(G') = \lambda_3(G) = 2$. However, this approach is manageable only if the number n of nodes and the number m of unconnected edges, i.e. $m = |\mathcal{E}^c|$,[1] in G are relatively small. Note that if the symmetric QR algorithm in [8] is used to compute eigenvalues, this approach runs in $O(4mn^3/3)$.

Another approach is the greedy perturbation heuristic proposed in [6]. Once the eigenvector (Fiedler vector) $\mathbf{v}_2$ associated with $\lambda_2(G)$ is obtained, the heuristic chooses an edge $e_{ij} \in \mathcal{E}^c$ with largest $(\mathbf{v_2}_i - \mathbf{v_2}_j)^2$, where $\mathbf{v_2}_i$ is the ith entry of $\mathbf{v_2}$. This approach is motivated by the fact that $(\mathbf{v_2}_i - \mathbf{v_2}_j)^2$ gives the first order approximation of the increase in $\lambda_2(G)$ (see [6] for more details). Clearly, this approach requires a lot less computational power than the first approach: its computational complexity is $O(2mn)$, excluding the once-off calculation of $\mathbf{v_2}$. The solution quality of the heuristic is generally good, as reported in [6] and will be shown in §III. We however note that there are still many cases in which the heuristic returns a poor solution. In fact, for the example in Figure 1, one obtains $e = e_{23}$ with $\lambda_2(G') = 1.5858$ using this heuristic.

B. Bisection algorithm

In the hope of finding an exact solution yet with small computational power, we propose a new algorithm in this section. To this end, consider the following modified version of the secular equation in [2]:

$$f(\lambda(G')) = 1 + \rho \sum_{\mathbf{k}=2}^{n} \frac{(\mathbf{v}_{\mathbf{k}i} - \mathbf{v}_{\mathbf{k}j})^2}{\lambda_{\mathbf{k}}(G) - \lambda(G')} = 0, \qquad (2.1)$$

where $\rho = 1$, G' is obtained by adding $e_{ij} \in \mathcal{E}^c$ to G, $\mathbf{v_k}$ is the eigenvector associated with the kth smallest eigenvalue of L_G, $\mathbf{v_k}_i$ is the ith entry of $\mathbf{v_k}$, and $\lambda(G')$ is an eigenvalue of $L_{G'}$. Note that $\rho = -1$ corresponds to the case where an edge is deleted from G. In general, the secular equation is used to find eigenvalues of diagonal plus rank-one matrices. As adding an edge to G may be viewed as a rank-one modification of L_G, the original secular equation can be written as (2.1) using a similarity transformation.[2]

If $\lambda_{\mathbf{k}}(G)$ and $\mathbf{v_k}$ for every $\mathbf{k}$ are known, the problem reduces to solving (2.1) for $\lambda(G')$, actually for $\lambda_2(G')$. As observed in [8], the derivative of $f(\lambda(G'))$ with respect to $\lambda(G')$ implies that $f(\lambda(G'))$ is monotonically increasing between $\lambda_2(G)$ and $\lambda_3(G)$. Since $\lambda_2(G) < \lambda_2(G') \leq \lambda_3(G)$, we thus have

$$f(\lambda(G')) \begin{cases} > 0 & \text{if} \quad \lambda(G') > \lambda_2(G'), \\ = 0 & \text{if} \quad \lambda(G') = \lambda_2(G'), \\ < 0 & \text{if} \quad \lambda(G') < \lambda_2(G'). \end{cases} \qquad (2.2)$$

From these characteristics of $f(\lambda_2(G'))$, one can deduce the following useful facts: (1) If $f(\lambda(G')) > 0$ for a certain $\lambda(G')$, the added edge e_{ij} to G gives rise to $\lambda_2(G') \in (\lambda_2(G), \lambda(G'))$; (2) If $f(\lambda(G')) < 0$ for a certain $\lambda(G')$, the added edge e_{ij} to G gives rise to $\lambda_2(G') \in (\lambda(G'), \lambda_3(G)]$. These two simple facts lead to our bisection algorithm in which $\lambda(G')$ (M in Table 1) is varied until the best edge ($\in \mathcal{S}_x$ in Table 1) is found.

The bisection algorithm presented in Table 1 essentially decreases the size of $\mathcal{S}_x$ as x increases. The size of $\mathcal{S}_x$ is initially m when $x = 0$, but becomes smaller whenever edges in $\mathcal{S}_x$ which satisfy $f(M) \leq 0$ are found. The final

TABLE I
BISECTION ALGORITHM

Input	$\lambda_2(G)$ and $\lambda_3(G)$ with $d = \lambda_3(G) - \lambda_2(G) > 0$, $\mathcal{E}$ with $\lvert\mathcal{E}^c\rvert = m$, and $\epsilon > 0$.
Step 0	$L \leftarrow \lambda_2(G)$, $U \leftarrow \lambda_3(G)$, $M \leftarrow (L+U)/2$, $x \leftarrow 0$, and $\mathcal{S}_x \leftarrow \mathcal{E}^c$.
Step 1	If $\lvert\mathcal{S}_x\rvert = 1$ or $x > \log_2(d/\epsilon)$, then go to **Output**; otherwise, $\mathcal{S}^+ \leftarrow \{\}$, $\mathcal{S}^- \leftarrow \{\}$, and go to **Step 2**.
Step 2	For each element $e_{ij} \in \mathcal{S}_x$, $\mathcal{S}^+ \leftarrow \mathcal{S}^+ \cup e_{ij}$ if $f(M) > 0$; $\mathcal{S}^- \leftarrow \mathcal{S}^- \cup e_{ij}$ otherwise.
Step 3	If $\lvert\mathcal{S}^-\rvert = 0$, $U \leftarrow M$ and $\mathcal{S}_{x+1} \leftarrow \mathcal{S}^+$; otherwise, $L \leftarrow M$ and $\mathcal{S}_{x+1} \leftarrow \mathcal{S}^-$.
Step 4	$x \leftarrow x + 1$ and go to **Step 1**.
Output	$\mathcal{S}_x$.

[1] For a set $\mathcal{S}$, $|\mathcal{S}|$ means the number of elements in $\mathcal{S}$.

[2] A generalized version of (2.1) is to be derived in a journal version of this paper.

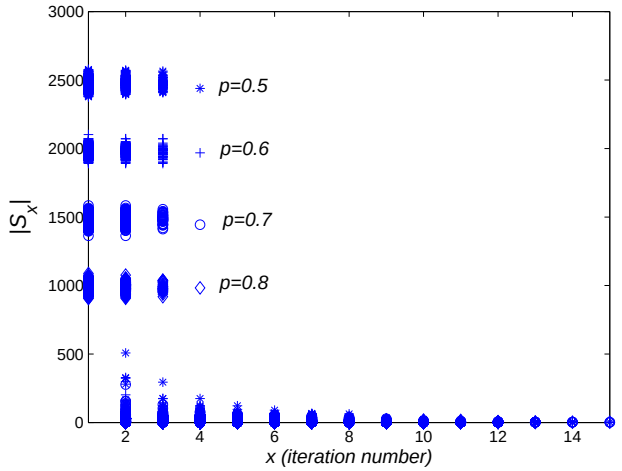


Fig. 2. $|\mathcal{S}_x|$ for four sets of 1000 random graphs with different p

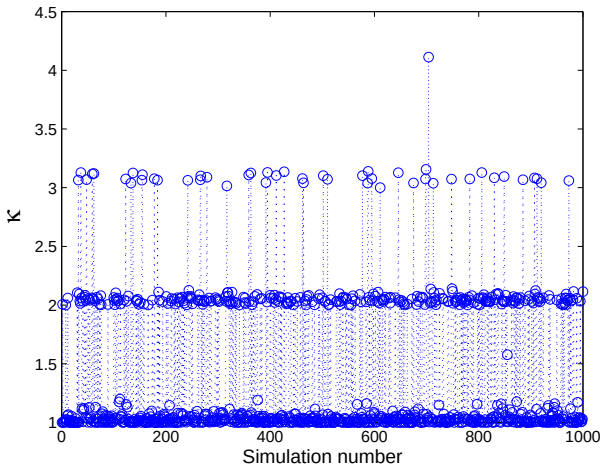


Fig. 3. κ for 1000 random graphs with $p = 0.7$

size of $\mathcal{S}_x$ must be 1 unless there exist two edges in $\mathcal{E}^c$ which yield the same or very close (less than ϵ) $\lambda_2(G')$. For the numerical examples in §III, we choose $\epsilon = 10^{-4}$. The termination criteria in **Step 1** are based on the following. The difference between the upper bound (U) and the lower bound (L) for some x is

$$|U - L| = \frac{1}{2^x}(\lambda_3(G) - \lambda_2(G)) = \frac{1}{2^x}d.$$

Thus, if one wishes to terminate the algorithm when $|U - L| < \epsilon$, then x must be greater than $\log_2(d/\epsilon)$.

The computational complexity of the proposed bisection algorithm depends on the size of $\mathcal{S}_x$ at each iteration. In view of $|\mathcal{S}_x| \leq m$ and (2.1), one can easily show that the algorithm runs in $O(4mn\log_2(d/\epsilon))$. However, we observe that as x increases, $|\mathcal{S}_x|$ decreases rapidly to 1 for most random graphs. For this observation, we create several sets of 1000 random graphs G with $\lambda_3(G) > \lambda_2(G) > 0$. The number of nodes in the random graphs is set to 100, and the edge between two nodes is added with a fixed probability of p. Figure 2 depicts $|\mathcal{S}_x|$ for four sets of the 1000 random graphs with different p. As shown in the figure, $|\mathcal{S}_x|$ decreases drastically for all the random graphs created, regardless of p.

For a more quantitative analysis, we calculate κ for each random graph such that

$$\sum_{x=1}^{\bar{x}} |\mathcal{S}_x| = \kappa m,$$

where $\bar{x}$ is the x when the algorithm is terminated. Figure 3 shows κ for 1000 random graphs with $p = 0.7$. The maximum but very much isolated κ is 4.112 and the average κ is 1.4352. This implies that the *average* or *practical* computational complexity of the proposed algorithm, $O(4\kappa mn) \approx O(5.7mn)$, is quite comparable to that of the simple greedy perturbation heuristic, $O(2mn)$.

Another remark is that our bisection algorithm does not allow $\lambda(G')$ to be $\lambda_i(G)$ $(i = 2, 3, \ldots, n)$, so no divide-by-zero error occurs when $f(\lambda_2(G))$ in (2.1) is evaluated. However, the algorithm can be successfully applied to the cases like the one shown in Figure 1, where $\lambda_2(G') = \lambda_3(G)$. This is because the algorithm does not find the exact *roots* of (2.1), but just find a best *edge*.

III. Comparison of Algorithms

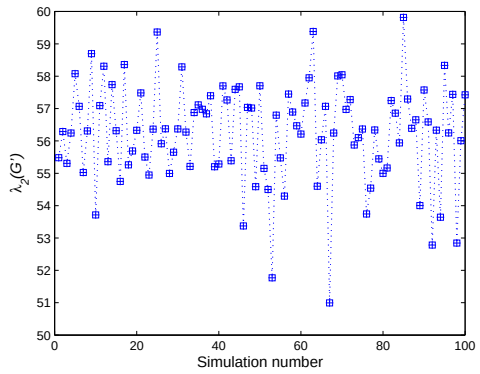
In this section, we compare the three algorithms: Alg I (first approach presented in §II-A), Alg II (second approach presented in §II-A) and Alg III (bisection algorithm in Table 1). For this purpose, we generate 100 random graphs with 100 nodes and $p = 0.7$ in the same manner as in the previous section.

As shown in Figure 4-(a), Alg III (cross) returns the exactly same $\lambda_2(G')$ as Alg I (square) does. However, Figure 4-(b) clearly suggests that Alg I requires much more computational power to do the same job. The solution quality of Alg II (circle) is generally good, as described in Figure 4-(c), except for several cases. Figure 4-(d) demonstrates that the computational times of Alg II and Alg III are quite close. This exactly supports the computational complexity analysis done in the previous section.

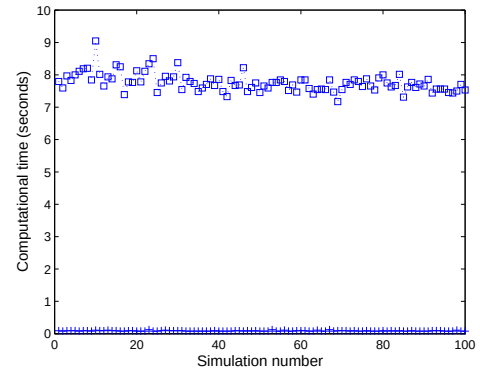
In conclusion, Alg III (proposed bisection algorithm) is indeed an efficient means of finding an edge such that $\lambda_2(G')$ is maximized by adding the edge.

IV. Conclusion

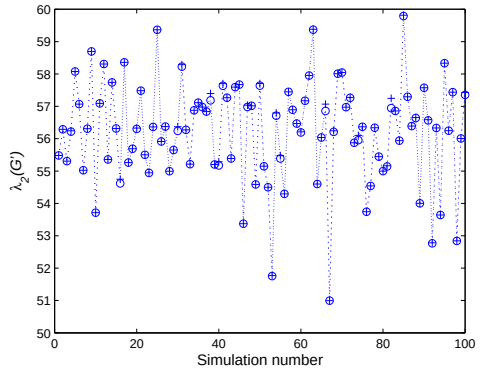
We presented a computationally efficient bisection algorithm of finding an edge which affects the algebraic connectivity of a graph G the most. The proposed algorithm makes use of a special property of the secular equation which gives eigenvalues of the modified graph G'. As a result, the proposed algorithm returns exact solutions but requires small computational power nearly comparable to the simple greedy perturbation heuristic. This merit was verified by extensive computer simulations.



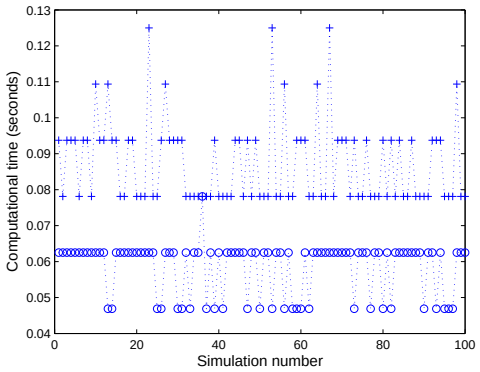
(a) $\lambda_2(G')$ obtained from Alg I (square) and Alg III (cross)



(b) Computational times of Alg I (square) and Alg III (cross)



(c) $\lambda_2(G')$ obtained from Alg II (circle) and Alg III (cross)



(d) Computational times of Alg II (circle) and Alg III (cross)

Fig. 4. Comparison of $\lambda_2(G')$ and computational times using 100 random graphs with $p = 0.7$

REFERENCES

[1] D. Barash. "Spectral Decomposition for the Search and Analysis of RNA Secondary Structure," *Journal of Computational Biology*, 11(6): 1169-1174, 2004.

[2] J. Bunch, C. Nielsen and D. Sorensen. "Rank one modification of the symmetric eigenproblem," *Numerical Mathematics*, vol. 31, no. 1, pp. 31–48, 1978.

[3] S. Fallat and S. Kirkland. "Extremizing algebraic connectivity subject to graph theoretic constraints," *The Electronic Journal of Linear Algebra*, vol. 3, no. 1, pp. 48–74, 1998.

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

[5] 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.

[6] A. Ghosh and S. Boyd. "Growing well-connected graphs," *Proceedings of the IEEE Conference on Decision and Control*, pp. 6605–6611, December 2006.

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

[8] G. Golub and C. F. Van Loan. *Matrix Computations*, The Johns Hopkins University Press, 1996.

[9] J. Guo. "The effect on the Laplacian spectral radius of a graph by adding or grafting edges," *Linear Algebra and its Applications*, vol. 413, no. 1, pp. 59-71, 2006.

[10] A. Hagberg and D. A. Schult. "Rewiring networks for synchronization," *Chaos*, vol. 18, pp. 037105, 2008.

[11] A.S. Ibrahim, K. G. Seddik, K. J. R Liu. "Improving Connectivity via Relays Deployment in Wireless Sensor Networks," *Proceedings of the IEEE Global Telecommunications Conference*, pp 1159–1163, November 2007.

[12] Y. Kim, D.-W. Gu and I. Postlethwaite. "Tight formation flying and sphere packing," *In Proceedings of American Control Conference*, pp. 1085–1090, July 2007.

[13] Y. Kim and M. Mesbahi. "On maximizing the second smallest eigenvalue of a state-dependent Laplacian," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 116–120, 2006.

[14] M. Mesbahi. "On state-dependent dynamic graphs and their controllability properties," *IEEE Transactions on Automatic Control*, vol. 50, no. 3, pp. 387–392, 2005.

[15] B. Mohar. "Some Applications of Laplace Eigenvalues of Graphs," *Graph Symmetry: Algebraic Methods and Applications*, Eds. G. Hahn and G. Sabidussi, NATO ASI Ser. C 497, Kluwer, pp. 225–275, 1997.

[16] R. Olfati-Saber and M. Murray. "Agreement problems in networks with directed graphs and switching topology," *Proceedings of the IEEE Conference on Decision and Control*, December 2003.

[17] T. Timothy. "Advances in sliding window subspace tracking," *PhD Thesis*, University of Rhode Island, 2005.