A spectral method to separate disconnected and nearly-disconnected Web graph components

Chris H.Q. Ding
NERSC Division
Lawrence Berkeley National Laboratory
University of California
Berkeley, CA 94720
chqding@lbl.gov

Xiaofeng He
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
xhe@cse.psu.edu

Hongyuan Zha
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
zha@cse.psu.edu

ABSTRACT

Separation of connected components from a graph with disconnected graph components mostly use breadth-first search (BFS) or depth-first search (DFS) graph algorithms. Here we propose a new algebraic method to separate disconnected and nearly-disconnected components. This method is based on spectral graph partitioning, following a key observation that disconnected components will show up, after properly sorted, as step-function like curve in the lowest eigenvectors of the Laplacian matrix of the graph. Following an perturbative analysis framework, we systematically analyzed the graph structures, first on the disconnected subgraph case, and second on the effects of adding edges sparsely connecting different subgraphs as a perturbation. Several new results are derived, providing insights to spectral methods and related clustering objective function. Examples are given illustrating the concepts and results our methods. Comparing to the standard graph algorithms, this method has the same $O(|E| + |V|\log(|V|))$ complexity, but is easier to implement (using readily available eigensolvers). Further more the method can easily identify articulation points and bridges on nearly-disconnected graphs. Segmentation of a real example of Web graph for query *amazon* is given. We found that each disconnected or nearly-disconnected components forms a cluster on a clear topic.

1. INTRODUCTION

Graph segmentation has wide range of applications. At one end is the near-regular graphs, the mesh of a 2D surface of an airfoil or 3D volumes of an engine cylinder, which are partitioned into subgraphs for a distributed memory parallel computations. At another end is the graphs generated from the World Wide Web. These graphs are highly irregular, and node degrees (both in-degrees and out-degrees) vary dramatically, from zero on millions of individual webpages to millions on a few popular websites.

Graphs generated from the retrieved Web pages of use queries or as the results of a web crawl often contain many disconnected and nearly-disconnected components. This fact is clearly shown in a large web structure analysis by Broder et al [4]. The disconnected components (called Weakly Connected Components there, because directions of links are ignored) follow power law distribution. In our small scale experiments, there is a large number of disconnected components in each retrieved web graph for a user query (see later section).

Separation of web graph into disconnected and nearly-disconnected components is very similar to clustering of web pages. Methods based on in-link and out-link relationship [18, 2], based max-flow [13], and based on text+link+co-citation [17] have been proposed to identify cluster or communities. Clustering of web pages helps to identify central topic for a query and therefore is very useful for web information retrieval and analysis. Besides clustering, there are also a large number of web link/connectivity information analyses [19, 24, 20, 22, 5].

For these reasons, separation of disconnected and nearly-disconnected components is useful to discover clusters of web communities and connectivity analysis, as an important application of a general method for graph segmentation problem.

Standard method to find disconnected components are breadth-first search (BFS) or depth-first-search (DFS) algorithms [8, 29]. BFS has been used in discovering the Strongly Connected Components (SCC, directions of hyperlinks are taken into account) and Weakly Connected Components [4]. A large number of graph algorithms for finding connected components exist [28, 1, 14, 15].

In this paper, we propose a new, algebraic method for finding disconnected and nearly-disconnected components, such as those illustrated in Figure 1. The method follows the spectral graph partitioning framework, first developed by Donath and Hoffman [10] and Fiedler [11, 12], and recently populated by the work of Pothen, Simon and Liu [25].

2. DISCONNECTED COMPONENTS

It is known [3, 11, 25] that the lowest eigenvalue of Lapla-

KDD '01 San Francisco, California, USA

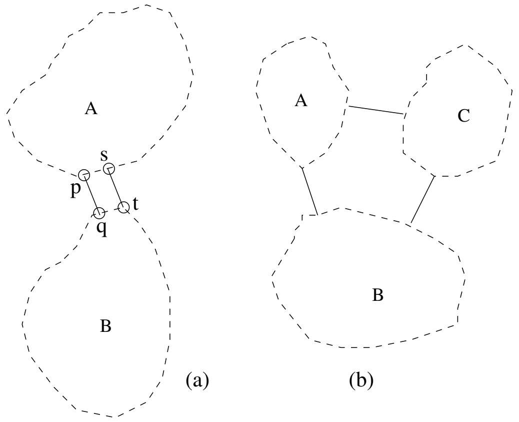

Figure 1: (a) Two dense subgraphs connected by two bridges (edges e_{pq}, e_{st}). Nodes p, q, s, t are called articulation points. (b) Three dense subgraphs connected by 3 bridges.

cian matrix of graph G has multiplicity of K if G has K disconnected components (more rigorously, K connected components of a disconnected graph). Our work is motivated by a key observation that disconnected components will show up as step-function like curve in the lowest eigenvectors, and thus can be easily identified. Furthermore, nearly-disconnected components (see Figure 1), will also show up as step-function like curve, although somewhat smoother. These new observations lead to a new class of *algebraic* algorithms to find these components.

Given an undirected weighted graph $G = (V, E)$ with adjacency (weight) matrix W, the Laplacian matrix is defined as $L = D - W$ where D is the diagonal matrix containing the node degrees. The above simple fact can be stated as:
Theorem. If a graph G has K disconnected components, then the lowest K eigenvalues of Laplacian matrix L are zero: $\lambda_i = 0,\ i = 1, \cdots, K$. Here $L\mathbf{y}_i = \lambda_i \mathbf{y}_i$.

We point out that the eigenvectors corresponding to the first K zero eigenvalues have step-function like curves if properly indexed, which can be used to identify the disconnected components. For simplicity, we assume G has $K = 2$ disconnected components. We index the nodes for subgraph $G_A = G_A(A, E_A)$ first and the nodes for subgraph $G_B = G_B(B, E_B)$ second. With this indexing, W has a block form,

$$W = \begin{pmatrix} W_{AA} & W_{AB} \\ W_{AB}^T & W_{BB} \end{pmatrix} \tag{1}$$

where W_{AB} contains the edges connecting the two subgraphs G_A, G_B. When G_A, G_B are disconnected, $W_{AB} = 0$, and the Laplacian matrix [1] has the block form:

$$L^{(0)} = \begin{pmatrix} D_{AA} - W_{AA} & 0 \\ 0 & D_{BB} - W_{BB} \end{pmatrix} \tag{2}$$

where $D_{AA} = \text{diag}(W_{AA}\mathbf{e}), D_{BB} = \text{diag}(W_{BB}\mathbf{e})$, where $\mathbf{e} = (1, \cdots, 1)^T$ with appropriate dimension. Let us define two indicator vectors:

$$\mathbf{e}^{(A)} = (1, \cdots, 1, 0, \cdots, 0)^T, \qquad \mathbf{x}^{(A)} = \mathbf{e}^{(A)}/\sqrt{|A|},$$
$$\mathbf{e}^{(B)} = (0, \cdots, 0, 1, \cdots, 1)^T, \qquad \mathbf{x}^{(B)} = \mathbf{e}^{(B)}/\sqrt{|B|},$$

for each subgraph (here $\mathbf{x}^{(A)}, \mathbf{x}^{(B)}$ are normalized to 1).

Clearly, $(D_{AA} - W_{AA})\mathbf{x}^{(A)} = 0$, and $(D_{BB} - W_{BB})\mathbf{x}^{(B)} = 0$. Thus L has two zero eigenvalues. In general, for any real a, b, the following vector

$$\mathbf{y} = a\mathbf{x}^{(A)} + b\mathbf{x}^{(B)}, \tag{3}$$

is an eigenvector of L with $\lambda = 0$: $L\mathbf{y} = 0$. Now, suppose we do not know whether graph G has 2 disconnected components. If we calculate the first eigenvector $\mathbf{v}_1$, and sort the nodal values and plot it, we will see a step-function, since from Eq.(4)

$$\mathbf{y}_1 \quad = \quad (a_1, \cdots, a_1, b_1, \cdots, b_1)^T \tag{4}$$

where $a_1 = a/\sqrt{|A|}, b_1 = b/\sqrt{|B|}$. If there are 3 disconnected components,

$$\mathbf{y} = a\mathbf{x}^{(A)} + b\mathbf{x}^{(B)} + c\mathbf{x}^{(C)},$$

we have $\mathbf{y}_1 = (a_1, \cdots, a_1, b_1, \cdots, b_1, c_1, \cdots, c_1)^T$, etc.

It is possible that two of the coefficients a, b, c, etc, are equal; in this case, the two components will be treated as one. Therefore, we need to check each identified component to see if it is a disconnected subgraph. Thus we have the following algorithm.

Algorithm DC (disconnected components).

1. Compute the lowest eigenvector $\mathbf{y}_1$ of L.
2. Sort $\mathbf{y}_1(i),\ i = 1, \cdots, n$. If $\mathbf{y}_1$ is a step function, there are disconnected components. Cut graph at the jump point i_{jump}. For each subgraph, repeat the same procedure.
3. If $\mathbf{y}_1$ is not a step function, i.e., $\mathbf{y}_1 = \mathbf{e}$, there is no disconnected components. Stop recursion.

Using Lanczos method, the calculation of the lowest eigenvalue and eigenvector pairs will converge very quickly [23]. A fast software for this calculation is available online (http://www.nersc.gov/~kewu/planso.html). The overall computation complexity for this algorithm is $O(|E| + |V| + |V|\log(|V|))$.

3. NEARLY-DISCONNECTED COMPONENTS

Many web subgraph components are nearly-disconnected, i.e., the dense subgraphs are sparsely connected only by a few bridges (edges) between them, as illustrated in Figure 1. The algebraic method can easily detect those bridges, whereas the standard graph algorithms can not easily do so.

Consider the situation of s bridges between two subgraphs A, B, as in Figure 1(a). In this case, W_{AB} is nonzero, and

$$s = s(A, B) = \sum_{i \in A, j \in B} W_{ij} \tag{5}$$

is also called the size of the cut: i.e., the number of bridges or cut edges (the weight on an edge can be any positive value).

[1] In this paper, L, W, D are $n \times n$ matrices; Bold face lower case letters represent vectors.

We are interested in finding the two lowest eigenvalues and eigenvectors of the following Laplacian matrix,

$$L = L^{(0)} + L^{(1)} \tag{6}$$

where $L^{(0)}$ is given by Eq.(3) for the disconnected components and $L^{(1)}$ is a small perturbation due to the s bridges:

$$L^{(1)} = \begin{pmatrix} D_{AB} & -W_{AB} \\ -W_{AB}^T & D_{AB} \end{pmatrix} \tag{7}$$

where $D_{AB} = \mathrm{diag}(W_{AB}\mathbf{e})$. Now we solve $L^{(1)}$ in the basis of $(\mathbf{x}^{(A)}, \mathbf{x}^{(B)})$, so $L^{(1)}$ is transformed into

$$\tilde{L}^{(1)} = (\mathbf{x}^{(A)}, \mathbf{x}^{(B)})^T L^{(1)} (\mathbf{x}^{(A)}, \mathbf{x}^{(B)})$$

and Eq.(3) is written as

$$\mathbf{y} = (\mathbf{x}^{(A)}, \mathbf{x}^{(B)}) \begin{pmatrix} a \\ b \end{pmatrix} = (\mathbf{x}^{(A)}, \mathbf{x}^{(B)}) \tilde{\mathbf{y}}^{(1)}$$

A little calculation gives

$$\tilde{L}^{(1)} = \begin{pmatrix} s/|A| & -s/\sqrt{|A||B|} \\ -s/\sqrt{|A||B|} & s/|B| \end{pmatrix}. \tag{8}$$

Equation $\tilde{L}^{(1)}\tilde{\mathbf{y}}^{(1)} = \lambda^{(1)}\tilde{\mathbf{y}}^{(1)}$ has two eigenvalues

$$\lambda_1^{(1)} = 0, \quad \lambda_2^{(1)} = \frac{s}{|A|} + \frac{s}{|B|}.$$

Thus the eigenvalues for entire problem are

$$\begin{aligned} \lambda_1 &= \lambda_1^{(0)} + \lambda_1^{(1)} = 0, \\ \lambda_2 &= \lambda_2^{(0)} + \lambda_2^{(1)} = \frac{s}{|A|} + \frac{s}{|B|} \equiv J_{\mathrm{rcut}}. \end{aligned} \tag{9}$$

The corresponding eigenvectors are

$$\tilde{\mathbf{y}}_1^{(1)} = \begin{pmatrix} \sqrt{|A|} \\ \sqrt{|B|} \end{pmatrix}, \quad \tilde{\mathbf{y}}_2^{(1)} = \begin{pmatrix} \sqrt{|B|} \\ -\sqrt{|A|} \end{pmatrix}. \tag{10}$$

Thus the eigenvectors for the entire problem are

$$\begin{aligned} \mathbf{y}_1 &= \sqrt{|A|}\mathbf{x}^{(A)} + \sqrt{|B|}\mathbf{x}^{(B)} = \mathbf{e} \\ \mathbf{y}_2 &= \sqrt{|B|}\mathbf{x}^{(A)} - \sqrt{|A|}\mathbf{x}^{(B)} = a_2\mathbf{e}^{(A)} - b_2\mathbf{e}^{(B)} \end{aligned}$$

where

$$a_2 = \sqrt{|B|/|A|}, \; b_2 = \sqrt{|A|/|B|} \tag{11}$$

within the accuracy of perturbation theory. (Details of this type of perturbation analysis can be found in [21, 26].)

Figure 2 shows the solution $\mathbf{y}_2$ for the graph as in Figure 1(a). The solution is very much like a step function, as we expected. The computed $\lambda_2 = 0.148$ (vs theoretical value of $J_{\mathrm{rcut}} = 0.166$).

It can be shown easily that $\mathbf{y}_2$ is the solution of minimizing the following function

$$J(\mathbf{y}) = \frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^T\mathbf{y}} = \frac{\sum_{ij}(y_i - y_j)^2 W_{ij}}{y^T y}. \tag{12}$$

subject to the condition $0 = \mathbf{y}^T\mathbf{y}_1 = \mathbf{y}^T\mathbf{e}$ and λ_2 is the minimum value. This implies that adjacent nodes will have close nodal values. Thus in Figure 1(a), nodal value y_p of articulation point p will be close to y_q on articulation point q. This means y_p will be pulled towards the nodal values of subgraph B. Similarly, y_q will be pulled towards the nodal values on subgraph A. These trends are clearly shown in Figure 2.

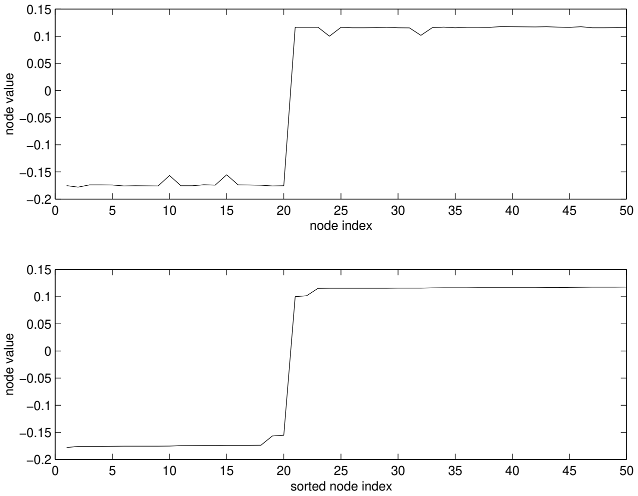

Figure 2: Computed $\mathbf{y}_2$ for the nearly-disconnected graph with two bridges. Two subgraphs have 20 and 30 nodes respectively. Top curve is $\mathbf{y}_2$ in original node index. Notice that node values on the four articulation points (nodes 10, 15, 24, 32) clearly differ from the rest in the same subgraph. Bottom curve is the sorted $\mathbf{y}_2$. Note that articulation points move to the jumping area, therefore could be easily identified.

For $K \geq 3$ multiple subgraphs, which are sparsely connected by a few bridges, the situation remains the same. After sorting, the nodal value plot of $\mathbf{y}_2$ will be a multi-step function, and the articulation points will move to the jump areas, and can be easily identified. Figure 3 shows the situation for 3-subgraph case.

Overall, the situations for nearly-disconnected components are one-to-one correspondence to those of disconnected components. The only difference is that the exact step function in the case of disconnected components become a smooth but sharp step-function, and the articulation points are located around the jumps.

With these analyses, the nearly-disconnected components can be detected by the following algorithm:

Algorithm NDC (nearly-disconnected components).

1. Compute the second lowest eigenvalue and eigenvector pair of L.

2. If λ_2 is small, but nonzero, there are nearly-disconnected components. Sort $\mathbf{y}_2$, cut graph at jump point i_{jump}. For each subgraph, repeat the same procedure.

3. If λ_2 is not small, i.e., $\lambda_2 \gg J_2$, there is no nearly-disconnected components. Stop recursion.

Algorithm NDC can be combined with algorithm DC to separate the components. Before discussing application of this algorithm to the web graph, we turn to the connection to spectral graph partitioning. As we will see, the results in this section is an explicit construction of Fiedler vector, in the near degenerate case when λ_2 is very close to zero.

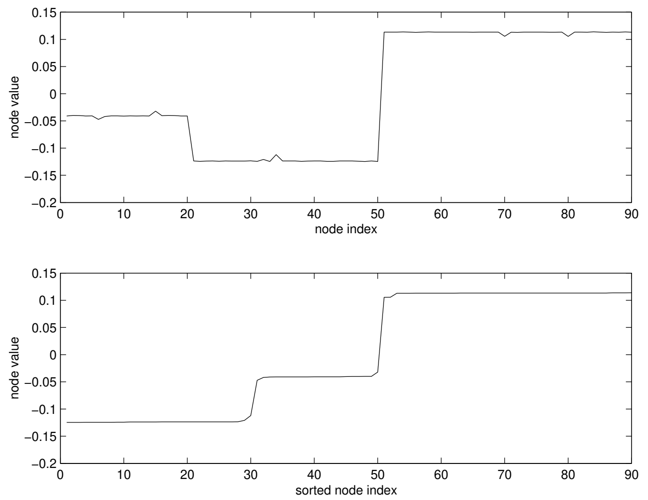

Figure 3: Computed $\mathbf{y}_2$ for 3 nearly disconnected subgraphs (see Figure 1(b)). After sorting $\mathbf{y}_2$, the six articulation points move to the jumping areas.

4. CONNECTION TO SPECTRAL GRAPH PARTITIONING

Algorithm NDC is an extension of spectral graph partitioning method in the case of nearly-disconnected components. The second lowest eigenvalue is called the Fiedler value and the eigenvector is called the Fiedler vector.

Our perturbative solution to Fielder value, Eq.(9), is identical to the *ratio cut* criteria [6, 16], [where they use the objective function $J = s/|A||B|$ which can be written as $J = (s/|A| + s/|B|)/|V|$ and therefore is the same as $J_{\rm rcut}$, since $|V| = |A| + |B|$ is fixed]. This new results further strengths the connection between ratio-cut and Fiedler vector of $(D - W)y = \lambda y$.

We could equally well start with the generalized eigenvector equation $(D - W)y = \lambda D\mathbf{y}$ and use associated eigenvector in our method. This equation can be written as

$$\widehat{L}\mathbf{z} = (I - D^{-1/2}WD^{-1/2})\mathbf{z} = \lambda\mathbf{z}$$

where $\mathbf{z} = D^{1/2}\mathbf{y}$. $\widehat{L}$ is called normalized Laplacian [7, 27]. When the graph has K disconnected components, $\widehat{L}$ will have block diagonal form, $\widehat{L}^{(0)} = \mathrm{diag}(\widehat{L}_{AA}, \widehat{L}_{BB}, \widehat{L}_{CC})$ for $K = 3$. Solutions to $\widehat{L}^{(0)}$, $\mathbf{x}^{(A)}, \mathbf{x}^{(B)}, \mathbf{x}^{(C)}$, can be similarly written down. When the K subgraphs are sparsely connected, these connections are contained in $\widehat{L}^{(1)}$, which can be viewed as a perturbation: $\widehat{L} = \widehat{L}^{(0)} + \widehat{L}^{(1)}$. Following through the similar analysis, we will get either

$$\lambda_2 = J_{\rm ncut} \equiv \frac{s(A,B)}{\deg(A)} + \frac{s(A,B)}{\deg(B)}$$

or

$$\lambda_2 = J_{\rm mcut} \equiv \frac{s(A,B)}{W(A)} + \frac{s(A,B)}{W(B)}$$

depending on how D is approximated. Here $\deg(A) = \sum_{i\in A} d_i$ is the sum over all node degrees in A and is called the degree of the subgraph A, and

$$W(A) = \sum_{i\in A}\sum_{j\in A} W_{ij}$$

is the sum of edge weights and node weights in A and is called the weight of the subgraph A. Note that $J_{\rm ncut}$ is the objective function for normalized cut criteria [27], and $J_{\rm mcut}$ is the objective function for min-max cut criteria [9].

5. WEB GRAPH SEGMENTATION

The web graph dataset is obtained in the following way [18]. We first submit a query to a search engine, which returns as query result a list of URLs with highest ranked webpages. We only take top 120 of returned URLs which form the *root set*. The root set is expanded into a *neighborhood set* of webpages by including all webpages that point to a webpage in the root set and those pointed to by a webpage in the root set. The hyperlinks between these webpages form a directed link graph, from which the adjacency matrix is extracted. The experiment dataset is retrieved for query *amazon*, with total 2294 webpages (URLs).

Using algorithm DC, we found the graph has 16 disconnected components, with 2181, 27, 18, ... nodes in each component respectively. This graph is dominated by a single component. It is interesting to note that at this small scale (2294 nodes), the distribution of component sizes is very similar to that of a very large web graph of Broder et al [4]. In their web graph of 200 million nodes, the largest weak connected component (WCC) has 186 million nodes, the second WCC has 60 thousands nodes, etc.

Below, to save space, we apply HITS algorithm [18] on each component (we sometimes call it cluster) to find the top authorities and list them. We first list the results on the 2nd and 3rd largest disconnected components (components A, B below), and discussed in details on separating the 1st largest 2181-node disconnected components into several nearly-disconnected sub-components (C,D,E,F,G,H).

Component A:

`www.internext.com.br/ariau`

This is a upscale hotel in the middle of Amazon region. This disconnected component has 27 webpages, where every other webpage points to this hotel webpage. Most webpages are on safaris, ecology, etc.

Component B:

```
www.latingrocer.com
www.latingrocer.com/Pages/customer.html
www.latingrocer.com/Pages/privacy.html
www.latingrocer.com/Pages/contact.html
```

This disconnected component contains all webpages on a single site.

When applying Algorithm NDC on the large disconnected component with 2181 nodes and separating it into nearly-disconnected sub-components, we get 4 sub-components:

Component C:

```
www.swalliance.com/
timeline.echostation.com
```

This component is about *Star Wars* alliance, a commercial site derived from the movie. The query word *amazon* happens to be mentioned in this webpage.

Component D:

```
www.amazoncity.com/
www.amazoncityradio.com/
```

www.amazoncity.com/spiderwoman/
www.wired.com/news/news/culture/...

This cluster is about female lesbian issue.

Component E:
misc.langenberg.com/
cooking.langenberg.com/
shipping.langenberg.com/
money.langenberg.com/
weather.langenberg.com/

This component contains all webpages on a single commercial site.

The 4th sub-component is a large one, with 1819 nodes. We applied DNC algorithm, separating it into 2 nearly-disconnected level 2 sub-components, cutting at node 1727, see sorted nodal values in Figure 4. Figure 5 shows the adjacency matrix with the sorted node index. One can clearly see that there are 3 bridges connecting the two nearly-disconnected sub-components. This illustrates that our algorithm can easily identify the hyperlinks between two nearly-disconnected components.

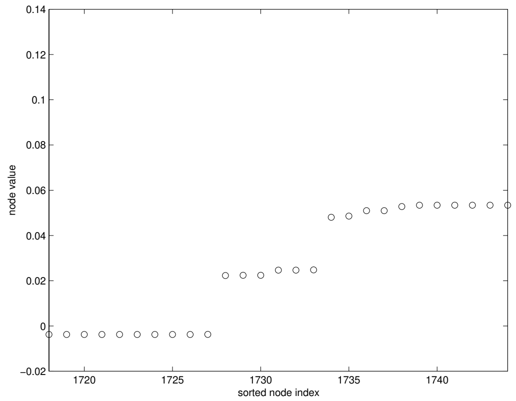

Figure 4: Details of nodal values of the Fiedler vector for the 1819-node component.

Based on sorted nodal values (Figure 5), we cut this component further into two sub- components, one large sub-component has 1727 nodes. The smaller sub-component has 92 nodes, among them the most informative nodes are

Component F:
www.amazon.org/
www.amazonfembks.com/
igc.apc.org/women/bookstores/
www.teleport.com/~rocky/queer.shtml
www.advocate.com/html/gaylinks/resources.html

This component is on female related issues: bi-sexuality and female books.

The 1727-node large sub-component can still be separated into two nearly-disconnected level 3 sub-components, with 7 bridges connecting these two components. They are

Component G:
www.amazon.com/
www.amazon.co.uk/
www.amazon.de

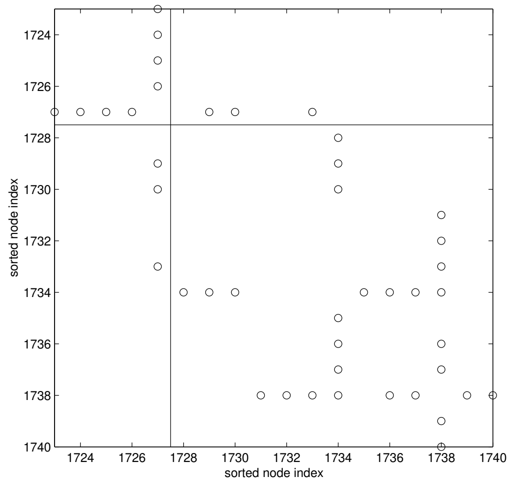

Figure 5: Details of the adjacency matrix of the 1819-node component when split into two nearly-disconnected sub-components (A in upper-left diagonal block and B in lower-right diagonal block). Node index are same as in Figure 4. Solid lines indicate the separator lines. The 3 points in the off-diagonal blocks represent 3 bridges connecting the two sub-components: node 1727 in A has three hyperlinks to nodes 1729, 1730, 1733 in subgraph B.

These three are the homepages of online bookstore amazon.com, and its subsidiaries in UK and in Germany.

Component H:
sothebys.amazon.com/exec/varzea/tg/special...
s1.amazon.com/exec/varzea/subst/home/home...
sothebys.amazon.com/exec/varzea/subst/home/...

All three authorities listed here are webpages of a large online auction company formed by Sothebys and amazon.com.

We note that all the components separated out by using connectivity information (adjacency matrix) only. These nearly same clusters are also identified using spectral partition based on normalized cut using on combined information of link connectivity, text and co-citation [17]. It appears that the connectivity information is the dominant factor.

6. SUMMARY AND DISCUSSIONS

Based on theoretical analysis, we proposed a new class of algebraic methods for separating densely connected components from a disconnected or nearly-disconnected graph. Although standard graph algorithms can readily separate disconnected components, they have difficulty to deal with nearly-disconnected components, whereas the algebraic methods deal with nearly-disconnected components in nearly identical way as with disconnected components, exhibiting some advantages over standard graph algorithms.

We provide several examples to illustrate some of the concepts and advantages of the proposed algorithms. We give a complete analysis of a web graph retrieved for a query word *amazon*. The separated components are found to be quite informative: each component represents a clear topic area.

We also discuss the relation to spectral graph partition and provided a new perspective on the objective function of the graph partitioning. In particular, from the perturbative analysis, we derived the ratio cut [6, 16], normalized cut [27], and min-max cut [9] objective functions from the (normalized) Laplacian matrix of a graph. The perturbation analysis framework, starting from disconnected subgraphs as diagonal blocks in the Laplacian matrix and adding sparse edge connections (bridges) between subgraphs as perturbations, provides a systematical way to analyze the structure of these graphs.

Identifying densely-connected components in nearly -disconnected graphs has a broad range of applications beyond the web graph analysis. It is also closely related to data clustering using a graph model, i.e., the similarity between data objects are represented by the graph weight matrix W, and the dense subgraphs are clusters. In fact, the dense components we found in the web graph are clearly web clusters (communities).

One issue in separating nearly-disconnected component is when to stop the recursive iterations. This is problem dependent. One may set a threshold for the cutsize $s(A, B)$ such as 3 bridges. Alternatively one may set a threshold for the objective function J_{rcut}, J_{ncut} or J_{mcut} as a clustering problem. We mention that J_{mcut} satisfies a min-max clustering principle — minimizing the similarity between clusters, $s(A, B)$, and maximizing the similarities within clusters $W(A)$, $W(B)$. In experiments, J_{mcut} appears to give more balanced clusters [9].

Acknowledgements. This work is supported in part by Office of Science, Office of Laboratory Policy and Infrastructure, of the U.S. Department of Energy under contract DE-AC03-76SF00098 through an LDRD grant in LBL.

7. REFERENCES

[1] B. Awerbuch and Y. Shiloach. New connectivity and msf algorithms for shuffle-exchange network and pram. *IEEE. Trans. on Computers*, pages 1258–1263, 1987.

[2] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. *ACM Conf. on Research and Development in Information Retrieval (SIGIR'98)*, 1998.

[3] N. Biggs. *Algebraic Graph Theory.* Cambridge Univ. Press, 1974.

[4] A. Broder, R Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Proc. 9th International World Wide Web Conference*, 2000.

[5] S. Chakrabarti, B. E. Dom, and J. M. Kleinberg. Mining the link structure of the world wide web. Feb 1999.

[6] C.-K. Cheng and Y.A. Wei. An improved two-way partitioning algorithm with stable performance. *IEEE. Trans. on Computed Aided Desgin*, 10:1502–1511, 1991.

[7] F.R.K. Chung. *Spectral Graph Theory.* Amer. Math. Society., 1997.

[8] T.H. Cormen, C.E. Leiserson, and R. L. Rives. *Introduction to Algorithms.* MIT Press, 1990.

[9] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. Spectral min-max cut for graph partitioning and data clustering. *Lawrence Berkeley Nat'l Lab Tech report 47848*, May 2001.

[10] W.E. Donath and A. J. Hoffman. Lower bounds for partitioning of graphs. *IBM J. Res. Develop.*, 17:420–425, 1973.

[11] M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, 23:298–305, 1973.

[12] M. Fiedler. A property of eigenvectors of non-negative symmetric matrices and its application to graph theory. *Czech. Math. J.*, 25:619–633, 1975.

[13] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. *Proc. Int'l Conf. Knowledge Kiscovery and Data Mining (KDD)*, pages 150–159, 2000.

[14] H. Gazit. An optimal randomized parallel algorithm for finding connected components. *SIAM J. Computing.*, 20:1046–1067, 1991.

[15] J. Griener. A comparison of data-parallel algorithms for connected components. *Proc. ACM Symp. Para. Algo. Arch.*, pages 16–25, 1994.

[16] L. Hagen and A.B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE. Trans. on Computed Aided Desgin*, 11:1074–1085, 1992.

[17] X. He, H. Zha, C. Ding, and H.D. Simon. Web document clustering using hyperlink structures. *Tech Report CSE-01-006*, April 2001.

[18] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 48:604–632, 1999.

[19] R. R. Larson. Bibliometrics of the world wide web: an exploratory analysis of the intellectual structures of cyberspace. *Proc. SIGIR'96*, 1996.

[20] Y. Li. Towards a qualitative search engine. *IEEE Internet Computing*, July-August 1998.

[21] J. Mathews and R.L. Walker. *Mathematical Methods of Physics.* Addison-Wesley, 1971.

[22] L. Page. Pagerank: bring order to the web. *Stanford Digital Library working paper 1997-0072*, 1997.

[23] B. N. Parlett. *The Symmetric Eigenvalue Problem.* SIAM Press, 1998.

[24] P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the web. *Proc. SIGCHI'96*, 1996.

[25] A. Pothen, H. D. Simon, and K. P. Liou. Partitioning sparse matrices with egenvectors of graph. *SIAM Journal of Matrix Anal. Appl.*, 11:430–452, 1990.

[26] L.I. Schiff. *Quantum Mechanics, 3rd ed.* McGraw-Hill, 1968.

[27] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 2000.

[28] Y. Shiloach and U. Vishkin. An o(log n) parallel connectivity algorithm. *J. Algorithm*, pages 57–67, 1982.

[29] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Computing.*, 1:146–160, 1972.