# Fast Spectral Clustering of Data Using Sequential Matrix Compression

Bo Chen[1,*], Bin Gao[2,*], Tie-Yan Liu[3], Yu-Fu Chen[1], and Wei-Ying Ma[3]

[1] Department of mathematics, Graduate School of Chinese Academic of Science
Beijing, 100080, P.R. China
chbo04@mails.gucas.ac.cn, yfchen@gucas.ac.cn
[2] LMAM, School of Mathematical Sciences, Peking University
Beijing, 100871, P.R. China
gaobin@math.pku.edu.cn
[3] Microsoft Research Asia, Sigma Center, No. 49, Zhichun Road,
Beijing, 100080, P.R. China
{tyliu, wyma}@microsoft.com

**Abstract.** Spectral clustering has attracted much research interest in recent years since it can yield impressively good clustering results. Traditional spectral clustering algorithms first solve an eigenvalue decomposition problem to get the low-dimensional embedding of the data points, and then apply some heuristic methods such as *k*-means to get the desired clusters. However, eigenvalue decomposition is very time-consuming, making the overall complexity of spectral clustering very high, and thus preventing spectral clustering from being widely applied in large-scale problems. To tackle this problem, different from traditional algorithms, we propose a very fast and scalable spectral clustering algorithm called the sequential matrix compression (SMC) method. In this algorithm, we scale down the computational complexity of spectral clustering by sequentially reducing the dimension of the Laplacian matrix in the iteration steps with very little loss of accuracy. Experiments showed the feasibility and efficiency of the proposed algorithm.

## 1 Introduction

Spectral clustering [3][6]is one of the most promising methods among existing clustering algorithms. Although a lot of previous work demonstrated the effectiveness of spectral clustering, its applications are mainly restricted to small-scale problems. This is due to the high computational complexity of spectral clustering. In the traditional implementation of spectral clustering, eigenvalue decomposition (EVD) is first conducted and then some additional heuristics such as *k*-means are applied to the eigenvectors to obtain the discrete cluster labels. It is known that the time and space complexities of state-of-the-art EVD solvers (such as Lanczos method [7] and pre-

[*] This work was performed when the first two authors were interns at Microsoft Research Asia.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): ECML 2006, LNAI 4212, pp. 590 – 597, 2006.

conditioned conjugate gradient (CG-based) algorithm [4][8]) are $O(mn^2k)$ and $O(n^2k)$ [2], where $k$ is the number of the eigenvectors used, $n$ is the number of data points, and $m$ is the number of iteration steps. It is clear such complexities are too high when the number of data points is large.

In order to extend spectral clustering to large-scale applications, we investigate how to reduce the complexity of spectral clustering in this paper. Our work is based on the following observation. When we compute the eigenvector associated with the second smallest eigenvalue in the EVD process of spectral clustering, we find that some specific elements in this eigenvector (also called the embedding vector or Fiedler vector) get stable very fast after only several steps of iteration and their values will not change by much in the future iteration steps. This observation indicates that it will not cause much loss if one stops the iteration process early for such stable points and fixes their values directly. It is easy to understand that this kind of early stop can reduce the problem scale of spectral clustering and save many computations.

However, one may argue that the embedding values of other data points will be affected if we manually fix the stable points because the optimizations of data points are not independent of each other. To tackle this problem, that is, to save the computations for the stable points but not to affect other points, we propose a matrix compression technology. Take two-way spectral clustering for example. After determining which stable points should be stopped early, we fix their values and merge those fixed data points with positive (negative) embedding values to a single aggregated positive (negative) point. Thus the scale of the Laplacian matrix is reduced. Then we properly adjust the values of the elements in this reduced Laplacian matrix, so that the embedding values of those unfixed points calculated from this reduced Laplacian matrix can be almost the same as their values calculated from the original Laplacian matrix. In this way, we may not only reduce the complexity of spectral clustering, but also successfully minimize the loss caused by the dimensionality reduction.

Actually the above idea can be once again applied to the reduced Laplacian matrix so that the problem scale can be further reduced. By conducting this process recursively, we can eventually get a very efficient implementation of spectral clustering. We name this new technique by sequential matrix compression (SMC). We proved in this paper that the SMC method can preserve enough information to guarantee the accuracy of the solutions. Experimental evaluations on real-world clustering problems showed the effectiveness and efficiency of the proposed SMC method.

## 2 Sequential Matrix Compression

In this section, we take the two-way ratio-cut spectral clustering for example to describe a fast implementation of spectral clustering based on sequential matrix compression. The same idea can also be extended to the normalized cut.

Suppose $L = \{l_{ij}\}$ is an $n \times n$ Laplacian matrix generated from a certain dataset and $\xi$ is the underlying $n$-dimensional embedding vector for clustering. That is, $L=D-W$, where $W$ is the adjacency matrix and $D$ is a diagonal matrix with the sum of each row of $W$ assigned to its corresponding diagonal positions. Then the two-way ratio-cut spectral clustering can be formulated as follows.

$$\min \xi^T L\xi \quad \textit{subject to} \;\; \xi^T\xi = 1,\; \xi^T e = 0. \tag{1}$$

It is clear that the above optimization problem is equivalent to finding the eigenvector associated with the second smallest eigenvalue of the following EVD problem.

$$L\xi = \lambda\xi. \tag{2}$$

Suppose we have found ($n$-$k$) stable elements after several steps of iteration, then the rows and columns of the Laplacian matrix $L$ can be re-organized as the matrix on the left-hand side of (3) shows so that the first ($n$-$k$) rows (columns) correspond to the stable elements (which are to be fixed in the subsequent iterations) and the rest $k$ rows (columns) correspond to the unfixed elements. As the Laplacian matrix is symmetrical, we have $L_{12}^T = L_{21}$. After applying the matrix compression strategy, the ($n$-$k$) stable elements are merged to an aggregated positive point and an aggregated negative one. Thus we can compress $L$ into a ($k$+2)×($k$+2) matrix $\hat{L} = \{\hat{l}_{ij}\}$ like the matrix on the right-hand side of (3). To keep $\hat{L}$ symmetrical, we let $\hat{L}_{12}^T = \hat{L}_{21}$. As the matrix compression strategy will not change the interrelations between the unfixed points, $L_{22}$ remains unchanged in $\hat{L}$.

$$\begin{array}{c} \\ L = \begin{matrix} n-k \\ k \end{matrix} \end{array} \begin{array}{c} \begin{matrix} n-k & k \end{matrix} \\ \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} \end{array} \quad \begin{array}{c} \textit{compress} \\ \Rightarrow \end{array} \quad \begin{array}{c} \\ \hat{L} = \begin{matrix} 2 \\ k \end{matrix} \end{array} \begin{array}{c} \begin{matrix} 2 & k \end{matrix} \\ \begin{bmatrix} \hat{L}_{11} & \hat{L}_{12} \\ \hat{L}_{21} & L_{22} \end{bmatrix} \end{array} \tag{3}$$

After the compression of the Laplacian matrix, the original spectral clustering problem (1) is converted to a smaller-scaled spectral clustering problem as follows.

$$\min \hat{\xi}^T \hat{L}\hat{\xi} \quad \textit{subject to} \;\; \hat{\xi}^T\hat{\xi} = 1,\; \hat{\xi}^T e = 0\,. \tag{4}$$

Here $\hat{\xi}$ is an ($k$+2)-dimensional embedding vector. It is natural that the solution to (4) is the eigenvector corresponding to the second smallest eigenvalue of the following EVD problem.

$$\hat{L}\hat{\xi} = \hat{\lambda}\hat{\xi}\,. \tag{5}$$

According to the block structures of $L$ and $\hat{L}$, we rewrite $\xi$ and $\hat{\xi}$ by $\xi = [\xi_1 \quad \xi_2]^T$ and $\hat{\xi} = \left[\hat{\xi}_1 \quad \hat{\xi}_2\right]^T$. Our objective is *to keep the second smallest eigenvalues of $L$ and $\hat{L}$ equal to each other, and to keep the embedding values of the unfixed elements calculated from $L$ and $\hat{L}$ exactly the same, i.e., to keep $\hat{\lambda} = \lambda$ and $\hat{\xi}_2 = \xi_2$*. For this purpose, we will investigate how to build the Laplacian matrix $\hat{L}$, i.e., how to determine $\hat{L}_{11}$, $\hat{L}_{12}$, and $\hat{L}_{21}$.

Letting $\hat{\xi}_2 = \xi_2$, from the constraints of the original optimization problem (1) and the new optimization problem (4), we can have the following equations.

$$\left.\begin{array}{ll} \xi_1^T\xi_1+\xi_2^T\xi_2=1, & \hat{\xi}_1^T\hat{\xi}_1+\xi_2^T\xi_2=1 \quad \Rightarrow \ \hat{\xi}_1^T\hat{\xi}_1=\xi_1^T\xi_1 \\ \xi_1^T e+\xi_2^T e=0, & \hat{\xi}_1^T e+\xi_2^T e=0 \quad \Rightarrow \quad \hat{\xi}_1^T e=\xi_1^T e \end{array}\right\}. \tag{6}$$

As we want to fix the stable elements in the embedding vector of the original problem to some discrete values, we can assume $\hat{\xi}_1=(a,-b)^T, a>0, b>0$, $\xi_1=(c_1,c_2,...,c_{n-k})^T$ and denote $d_1=\xi_1^T\xi_1=\sum_{i=1}^{n-k}c_i^2$, $d_2=\xi_1^T e=\sum_{i=1}^{n-k}c_i$. Then by substituting the above notations to (6) and solving the corresponding equation set, we have

$$a=\left(d_2+\sqrt{2d_1-d_2^2}\right)\Big/2,\ b=\left(-d_2+\sqrt{2d_1-d_2^2}\right)\Big/2. \tag{7}$$

Therefore, we can compute $\hat{\xi}_1$ with (7) based on $\xi_1$, as a necessary condition to guarantee $\hat{\xi}_2=\xi_2$.

Decomposing the matrices in the EVD equations (2) and (5) into blocks, and considering $\hat{\lambda}=\lambda$ and $\hat{\xi}_2=\xi_2$, we can get

$$\begin{array}{llll} L_{11}\xi_1+L_{12}\xi_2=\lambda\xi_1 & (i); & L_{21}\xi_1+L_{22}\xi_2=\lambda\xi_2 & (ii); \\ \hat{L}_{11}\hat{\xi}_1+\hat{L}_{12}\xi_2=\lambda\hat{\xi}_1 & (iii); & \hat{L}_{21}\hat{\xi}_1+L_{22}\xi_2=\lambda\xi_2 & (iv). \end{array} \tag{8}$$

From (8)(*ii*) and (8)(*iv*), we have

$$\hat{L}_{21}\hat{\xi}_1=L_{21}\xi_1. \tag{9}$$

With the notations $\hat{\xi}_1, \xi_1, d_1, d_2$ defined above, we can rewrite equation (9) as below.

$$a\hat{l}_{i1}-b\hat{l}_{i2}=\sum_{j=1}^{n-k}c_j l_{(i+n-k-2),j},\ (i=3,...,k+2). \tag{10}$$

The sum of each row (column) of a Laplacian matrix should be zero. Therefore, for the last *k* rows of the two Laplacian matrices $L$ and $\hat{L}$, we have

$$\hat{l}_{i1}+\hat{l}_{i2}=\sum_{j=1}^{n-k}l_{(i+n-k-2),j},\ (i=3,...,k+2). \tag{11}$$

By solving the equation set as shown in (10) and (11), we can eventually get

$$\begin{cases} \hat{l}_{i1}=\left[\sum_{j=1}^{n-k}(b+c_j)l_{(i+n-k-2),j}\right]\Big/(a+b) \\ \hat{l}_{i2}=\left[\sum_{j=1}^{n-k}(a-c_j)l_{(i+n-k-2),j}\right]\Big/(a+b) \end{cases},\quad (i=3,...,k+2)\cdot \tag{12}$$

From (12), we can see that the weights between the two aggregated points and the unfixed points are the linear combinations of the weights between the original fixed points and the unfixed points. In other words, given $\xi_1$ and the Laplacian matrix *L*, we can calculate $\hat{L}_{12}$ and $\hat{L}_{21}$ using (12) with very little computational cost.

In the next step, we will discuss how to compute $\hat{L}_{11}$. According to the property of the Laplacian matrix, the sums of the first two columns of $\hat{L}$ should also be zeros. As a result, we have,

$$\begin{cases} \hat{l}_{11}+\hat{l}_{21}=-\sum_{i=3}^{k+2}\hat{l}_{i1}=-\left[\sum_{i=n-k+1}^{n}\sum_{j=1}^{n-k}(b+c_j)l_{ij}\right]\Big/(a+b)=-\left[be^TL_{21}e+e^TL_{21}\xi_1\right]\Big/(a+b) \\ \hat{l}_{12}+\hat{l}_{22}=-\sum_{i=3}^{k+2}\hat{l}_{i2}=-\left[\sum_{i=n-k+1}^{n}\sum_{j=1}^{n-k}(a-c_j)l_{ij}\right]\Big/(a+b)=-\left[ae^TL_{21}e-e^TL_{21}\xi_1\right]\Big/(a+b). \\ \hat{l}_{21}=\hat{l}_{12} \end{cases} \tag{13}$$

There are four unknown quantities and three equations in (13), so we have to find another equation to work out the unique solution for this equation set. This additional equation comes from the objective functions of the two optimization problems. Since we would like to keep the second smallest eigenvalues of (2) and (5) equal to each other, we can get the following equation.

$$\xi^TL\xi=\hat{\xi}^T\hat{L}\hat{\xi}. \tag{14}$$

Decomposing the matrices and vectors in (14) into blocks, and considering (9), we can get the following equation.

$$\xi_1^TL_{11}\xi_1=\hat{\xi}_1^T\hat{L}_{11}\hat{\xi}_1. \tag{15}$$

According to the notations $\hat{\xi}_1$, $\xi_1, d_1, d_2$ defined above, equation (15) is equivalent to

$$a^2\hat{l}_{11}-2ab(\hat{l}_{12}+\hat{l}_{21})+b^2\hat{l}_{22}=\xi_1^TL_{11}\xi_1. \tag{16}$$

Adding (16) to the equation set (13), we eventually have a solvable equation set which solution is shown as follows.

$$\begin{cases} \hat{l}_{11}=\left[\xi_1^TL_{11}\xi_1-2be^TL_{21}\xi_1-b^2e^TL_{21}e\right]\Big/(a+b)^2 \\ \hat{l}_{12}=\hat{l}_{21}=-\left[\xi_1^TL_{11}\xi_1+(a-b)e^TL_{21}\xi_1+abe^TL_{21}e\right]\Big/(a+b)^2 \\ \hat{l}_{22}=\left[\xi_1^TL_{11}\xi_1+2ae^TL_{21}\xi_1-a^2e^TL_{21}e\right]\Big/(a+b)^2 \end{cases}. \tag{17}$$

Up to now, we have successfully computed all the elements in the reduced Laplacian matrix $\hat{L}$ based on $\xi_1$ and $L$. Actually it is easy to understand what we get is not only a necessary condition but also a sufficient condition for $\hat{\xi}_2=\xi_2$. That is, if $\xi_1$ is precise, we can exactly have $\hat{\xi}_2=\xi_2$. In other words, even if we merge some stable points using the sequential matrix compression strategy, the embedding vector of the unfixed points will not be influenced. Therefore, we have derived a lossless method to scale down the computation cost of spectral clustering problems. We summarized this method as the ratio-cut **S**equential **M**atrix **C**ompression (SMC) algorithm in Table 1.

As for the initialization, we load the Laplacian matrix of the original optimization problem and compute its eigenvector associated with the second smallest eigenvalue by a certain EVD solver. After a certain number of iteration steps, we break off the

EVD solver and check the current status of the eigenvector $\xi^{(t)}$ in order to find out some stable elements (denoted by the sub-vector $\xi_1^{(t)}$). From $\xi_1^{(t)}$, the aggregated sub-vector $\hat{\xi}_1^{(t)}$ can be calculated by (7). Then the elements of the compressed Laplacian matrix $\hat{L}$ can be computed by (12) and (17). If the scale of matrix $\hat{L}$ is still large, another round of matrix compression might be conducted; otherwise, the EVD problem is worked out directly and the clustering result is generated accordingly.

**Table 1.** The SMC method

1. Set $t = 0$, and input the original Laplacian matrix $L^{(t)}$.
2. Compute the eigenvector according to the second smallest eigenvalue of $L^{(t)}$ by CG-based EVD solver, and break off the process after a certain number of iteration steps.
3. From the current status of the eigenvector $\xi^{(t)}$, select a vector $\xi_1^{(t)}$ whose elements are regarded as stable points. At the same time, $L_{22}^{(t)}$ is obtained and $L_{22}^{(t+1)} = L_{22}^{(t)}$.
4. Compute $\hat{\xi}_1^{(t)} = (a^{(t)}, -b^{(t)})^T$ by (7).
5. Compute $\hat{l}_{i1}^{(t)}, \hat{l}_{i2}^{(t)}$, $(i = 3, ..., k^{(t)} + 2)$ by (12) so that $L_{21}^{(t+1)}$ is obtained and $L_{12}^{(t+1)} = (L_{21}^{(t+1)})^T$.
6. Compute $\hat{l}_{11}^{(t)}, \hat{l}_{12}^{(t)}, \hat{l}_{21}^{(t)}, \hat{l}_{22}^{(t)}$ by (17) to obtain $L_{11}^{(t+1)}$.
7. Build the compressed Laplacian matrix $L^{(t+1)} = \begin{bmatrix} L_{11}^{(t+1)} & L_{12}^{(t+1)} \\ L_{21}^{(t+1)} & L_{22}^{(t+1)} \end{bmatrix}$ and go to Step 2 if the scale of $L^{(t+1)}$ is still large; otherwise, solve the eigenvalue problem of $L^{(t+1)}$ to get the corresponding eigenvector, and output the embedding vector after some necessary post-processing.

It is easy to get that the overall complexity of our proposed SMC method is $O(kn^2)$, where $k$ is the initial iteration steps (usually less than ten), and $n$ is the scale of the original Laplacian matrix. For comparison, the complexity of CG-based EVD solver is $O(mn^2)$, where $m$ is the total iteration steps (usually several hundred or even larger); while Lanczos-based EVD solver takes even more computational burden than CG-based EVD solver.

Note that after an error bound analysis, we can prove that the SMC algorithm is almost lossless. Moreover, the idea of sequential matrix compression can be easily extended to adapt the case of normalized cut, and most of the derivations are quite similar to those of the ratio cut. We omitted the above content due to the space limitation.

## 3 Experiments

In this section, we report the experiments that we conducted to show the efficiency and effectiveness of the proposed SMC algorithm. Considering that CG-based and

Lanczos-based EVD solvers are among the most popular and efficient algorithms for the EVD of sparse matrices, we use them as the baselines in our experiments. And since our theoretical derivations of the SMC algorithm is based on two-way clustering, all the experiments are also designed for two-way clustering.

When implementing our SMC method, CG-based EVD solver was adopted to compute the embedding vector of the optimization problem (1). After several steps of iteration, we used the following simple strategy to extract the stable sub-vector $\xi_1$. Suppose there are $n_1$ positive points and $n_2$ negative points in $\xi$. Then the top $p\%$ of the positive points in $\xi$ ($n_1p\%$ in number) were regarded as the positive working set, while the bottom $p\%$ of the negative points in $\xi$ ($n_2p\%$ in number) were regarded as the negative working set. Here $p\%$ is referred to as the proportion of fixed points. In the working sets, if the absolute value of an element's gradient was smaller than a very small threshold (e.g., 0.001.), this element would be regarded as a stable point. All positive stable points were merged to a new aggregated positive point in the next step iteration. Similarly, all negative stable points were merged to a new aggregated negative point. This process can be conducted in a recursive manner until all the data points are merged into two points, which indicates that all the points have been clustered into either of the two clusters.

We ran the SMC algorithm and the reference algorithms on the 20-newsgroups [5] dataset. Each document was represented by a feature vector of term frequency [1], and the weights in the adjacency matrix were calculated by (18), where $v_i$ is the feature vector of the corresponding document.

$$W(i,j) = v_i^T v_j \Big/ \left( \|v_i\|_2 \|v_j\|_2 \right) \tag{18}$$

**Table 2.** Average clustering error rate and average time cost for 20-newsgroups dataset

| Category name | Average clustering error rate | | | Average time cost | | |
|---|---|---|---|---|---|---|
| | CG | Lanczos | SMC | CG | Lanczos | SMC |
| *alt.atheism* | 0.027 | **0.026** | 0.039 | 1.325 | 5.440 | **0.888** |
| *comp.graphics* | 0.034 | **0.005** | 0.014 | 1.299 | 5.513 | **0.881** |
| *comp.os.ms-windows.misc* | 0.028 | **0.011** | 0.057 | 1.321 | 5.535 | **0.902** |
| *comp.sys.ibm.pc.hardware* | 0.023 | **0.006** | 0.020 | 1.173 | 5.509 | **0.903** |
| *comp.sys.mac.hardware* | 0.009 | **0.005** | 0.015 | 1.304 | 5.498 | **0.895** |
| *comp.windows.x.* | **0.005** | 0.010 | 0.022 | 1.280 | 5.600 | **0.887** |
| *misc.forsale* | **0.003** | 0.004 | 0.012 | 1.199 | 5.518 | **0.887** |
| *rec.autos* | **0.004** | 0.006 | 0.038 | 1.251 | 5.448 | **0.896** |
| *rec.motorcycles* | 0.008 | **0.003** | 0.010 | 1.265 | 5.455 | **0.887** |
| *rec.sport.baseball* | 0.005 | **0.003** | 0.039 | 1.279 | 5.450 | **0.887** |
| *rec.sport.hockey* | 0.006 | **0.002** | 0.012 | 1.291 | 5.476 | **0.897** |
| *sci.crypt* | 0.011 | **0.006** | 0.007 | 1.261 | 5.470 | **0.900** |
| *sci.electronics* | 0.043 | **0.015** | 0.032 | 1.376 | 5.501 | **0.905** |
| *sci.med* | **0.007** | 0.009 | 0.028 | 1.300 | 5.443 | **0.895** |
| *sci.space* | 0.040 | **0.004** | 0.048 | 1.451 | 5.435 | **0.902** |
| *soc.religion.christian* | 0.077 | **0.001** | 0.006 | 1.382 | 5.440 | **0.892** |
| *talk.politics.guns* | 0.044 | **0.024** | 0.058 | 1.386 | 5.435 | **0.901** |
| *talk.politics.mideast* | **0.051** | 0.058 | 0.079 | 1.290 | 5.455 | **0.898** |
| *talk.politics.misc* | 0.058 | **0.034** | 0.054 | 1.367 | 5.440 | **0.889** |
| *talk.religion.misc* | 0.062 | **0.041** | 0.057 | 1.291 | 5.472 | **0.894** |
| **Average** | 0.027 | **0.014** | 0.032 | 1.305 | 5.477 | **0.894** |

We tested our algorithm and the reference algorithms on every possible pair of categories in the 20-newsgroups dataset. The average performance for between each category and all the other categories are listed in Table 2, where the surpassing values are blackened. For average clustering error rate, we can see that the proposed SMC algorithm performed slightly worse than the reference algorithms. This is reasonable because we used an approximation of $\xi_1$ instead of the precise vector when compressing the Laplacian matrix. However, on average, the clustering error rates of all these algorithms were all very low and the differences between them were negligible. For average time cost, we can see that the SMC algorithm defeated the other two methods by much. This verified that our algorithm could depress the time complexity of spectral clustering. Overall speaking, the SMC algorithm can achieve higher efficiency with very little accuracy loss compared with the reference algorithms.

## 4 Conclusions

In this paper, we proposed a sequential matrix compression strategy to accelerate spectral clustering in order to fit the need of large-scale applications. The basic idea is to sequentially depress the scale of the Laplacian matrix in the iteration steps of spectral clustering. Experiments showed the efficiency and feasibility of our method.

## References

[1] Baeza-Yates, R., and Ribeiro-Neto, B. Modern information retrieval. ACM Press, a Division of the Association for Computing Machinery, Inc. (ACM). 1999.
[2] Golub, G.H., and Loan, C.F.V. Matrix computations. Johns Hopking University Press, 3rd edition, 1996.
[3] Hagen, L., and Kahng, A.B. New spectral methods for ratio cut partitioning and clustering. IEEE Transactions on CAD, 11:1074-1085, 1992.
[4] Knyazev, A.V. Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method. SIAM Journal on Scientific Computing, Volume 23, Number 2, pp. 517-541, 2001.
[5] Lang, K. News Weeder: Learning to filter netnews. In ICML-95, pp. 331-339, 1995.
[6] Shi, J., and Malik, J. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):888-905, August 2000.
[7] Sorensen, D.C. Implicitly-restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations. Technical Report TR-96-40, 1996.
[8] Yang, X.P., Sarkar, TK., Arvas, E. A Survey of Conjugate Gradient Algorithms for Solution of Extreme Eigen-problems of a Symmetric Matrix. IEEE Transactions on Acoustics Speech and Singal Processing, Vol 37, NO. 1000, pp.1550-1555, 1989.