A Fast Kernel-based Multilevel Algorithm for Graph Clustering

Inderjit Dhillon
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
inderjit@cs.utexas.edu

Yuqiang Guan
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
yguan@cs.utexas.edu

Brian Kulis
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
kulis@cs.utexas.edu

ABSTRACT

Graph clustering (also called graph partitioning) — clustering the nodes of a graph — is an important problem in diverse data mining applications. Traditional approaches involve optimization of graph clustering objectives such as normalized cut or ratio association; spectral methods are widely used for these objectives, but they require eigenvector computation which can be slow. Recently, graph clustering with a general cut objective has been shown to be mathematically equivalent to an appropriate weighted kernel k-means objective function. In this paper, we exploit this equivalence to develop a very fast multilevel algorithm for graph clustering. Multilevel approaches involve coarsening, initial partitioning and refinement phases, all of which may be specialized to different graph clustering objectives. Unlike existing multilevel clustering approaches, such as METIS, our algorithm does not constrain the cluster sizes to be nearly equal. Our approach gives a theoretical guarantee that the refinement step decreases the graph cut objective under consideration. Experiments show that we achieve better final objective function values as compared to a state-of-the-art spectral clustering algorithm: on a series of benchmark test graphs with up to thirty thousand nodes and one million edges, our algorithm achieves lower normalized cut values in 67% of our experiments and higher ratio association values in 100% of our experiments. Furthermore, on large graphs, our algorithm is significantly faster than spectral methods. Finally, our algorithm requires far less memory than spectral methods; we cluster a 1.2 million node movie network into 5000 clusters, which due to memory requirements cannot be done directly with spectral methods.

Categories and Subject Descriptors

G.1.8.1 [**Numerical Analysis**]: Spectral Methods; H.3.3.a [**Information Search and Retrieval**]: Clustering; I.5.3.a [**Pattern Recognition**]: Clustering Algorithms

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.

General Terms

Algorithms, Experimentation

Keywords

Graph Clustering, Kernel Methods, Multilevel Methods, Spectral Clustering

1. INTRODUCTION

Graph clustering (also called graph partitioning) is an important problem in many domains. Circuit partitioning, VLSI design, task scheduling, bioinformatics, social network analysis and a host of other problems all rely on efficient and effective graph clustering algorithms. In many data mining applications, pairwise similarities between data objects can be modeled as a graph, and the subsequent problem of data clustering can be viewed as a graph clustering problem.

The problem of graph clustering has been studied for decades, and a number of different approaches have been proposed. Spectral methods have been widely used for graph clustering [2, 6, 9]. These algorithms use the eigenvectors of a graph affinity matrix, or a matrix derived from the affinity matrix, to partition the graph. Various spectral algorithms have been developed for a number of different objective functions, such as normalized cut [9] and ratio cut [2]. Furthermore, extensive research has been done on spectral postprocessing: going from the eigenvector matrix to a discrete clustering [4, 10].

It was recently shown that a wide class of graph clustering objectives, including ratio cut, ratio association, the Kernighan-Lin objective and normalized cut, can all be viewed as special cases of the weighted kernel k-means objective function [3, 4]. In addition to unifying several different graph clustering objectives, including a number of spectral clustering objectives, this result implies that a simple weighted kernel k-means algorithm can be used to optimize graph clustering objectives. The basic kernel k-means algorithm, however, relies heavily on effective cluster initialization to achieve good results, and can often yield poor results.

In this paper, we develop a multilevel algorithm for graph clustering that uses weighted kernel k-means as the refinement algorithm. Multilevel methods have been used extensively for graph clustering with the Kernighan-Lin objective, which attempts to minimize the cut in the graph while maintaining equal-sized clusters [1, 7, 8]. In multilevel algorithms, the graph is repeatedly coarsened level by level until only a small number of nodes are left. Then, an ini-

Name	Objective Function
Ratio Association	$\underset{\mathcal{V}_1,\ldots,\mathcal{V}_k}{\text{maximize}} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c,\mathcal{V}_c)}{\lvert\mathcal{V}_c\rvert}$
Ratio Cut	$\underset{\mathcal{V}_1,\ldots,\mathcal{V}_k}{\text{minimize}} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c,\mathcal{V}\backslash\mathcal{V}_c)}{\lvert\mathcal{V}_c\rvert}$
Normalized Cut	$\underset{\mathcal{V}_1,\ldots,\mathcal{V}_k}{\text{minimize}} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c,\mathcal{V}\backslash\mathcal{V}_c)}{\text{degree}(\mathcal{V}_c)}$

Table 1: Examples of graph clustering objectives. Note that Normalized Association = $k-$ Normalized Cut.

tial clustering on this small graph is performed. Finally, the graph is uncoarsened level by level, and at each level, the clustering from the previous level is refined using the refinement algorithm. This multilevel strategy results in a very fast and effective graph clustering algorithm. METIS [8] is perhaps the most popular multilevel algorithm, and is significantly faster than spectral methods on very large graphs when equally sized clusters are desired.

However, all previous multilevel algorithms, such as METIS, suffer from the serious drawback of restricting clusters to be of equal size. The algorithm presented in this paper removes this restriction: instead of a Kernighan-Lin algorithm for the refinement phase, our algorithm employs weighted kernel k-means.

We present experimental results on a number of test graphs to illustrate the advantage of our approach. We show that our algorithm is significantly faster than eigenvector computation, a necessary step for spectral clustering methods, on large graphs. We also demonstrate that our algorithm outperforms a state-of-the-art spectral clustering algorithm in terms of quality, as measured by the graph objective function values. On our benchmark test graphs of varying sizes, the final objective function values of the multilevel algorithm are better than the spectral algorithm in 100% of runs for ratio association, and 67% of runs for normalized cut. We also present results on data arising from real-life data mining applications. We cluster the 1.2 million node IMDB movie network into 5000 clusters. Running a spectral algorithm directly on this data set would require storage of 5000 eigenvectors, which is prohibitive.

2. GRAPH CLUSTERING

We first introduce a number different graph clustering objectives. As we will see later, each of the following objectives may be expressed as special cases of the weighted kernel k-means objective function, which will motivate our use of weighted kernel k-means as the algorithm used in the refinement phase.

For data given as a graph, a number of different objectives have been proposed for the graph clustering problem. We are given a graph $G = (\mathcal{V}, \mathcal{E}, A)$, consisting of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$ such that an edge between two vertices represents their similarity. The affinity matrix A is $|\mathcal{V}| \times |\mathcal{V}|$ whose entries represent the weights of the edges (an entry of A is 0 if there is no edge between the corresponding vertices).

We denote $\text{links}(\mathcal{A}, \mathcal{B}) = \sum_{i\in\mathcal{A}, j\in\mathcal{B}} A_{ij}$; that is, the sum of the weights of edges from one cluster to the other. Let $\text{degree}(\mathcal{A}) = \text{links}(\mathcal{A}, \mathcal{V})$, the links to all vertices from nodes in $\mathcal{A}$.

A list of some of the most common graph clustering objectives is presented in Table 1. The Kernighan-Lin objective, another common objective function, is the same as ratio cut except that clusters are additionally constrained to be of equal size.

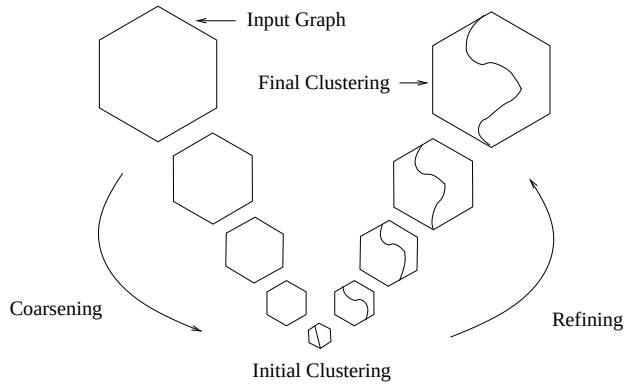

Figure 1: Overview of the multilevel algorithm (for $k = 2$).

For ratio association, ratio cut and normalized cut, the most common approach for optimization has been to use a spectral algorithm. Such an algorithm is derived by relaxing the above objectives so that a global solution to the relaxed objective may be found by computing eigenvectors of an appropriate matrix. Once these eigenvectors are computed, a postprocessing step derives a discrete clustering from the eigenvectors. See the references [2, 9, 10] for further details on the spectral algorithms associated with each of these objectives.

For the Kernighan-Lin objective, the most successful approach for optimizing the objective has been to use a multilevel algorithm.

3. THE MULTILEVEL APPROACH

In this section, we develop a new, general algorithm for graph clustering based on multilevel methods. The multilevel approach has been made popular by METIS [8], though a number of multilevel clustering algorithms have been studied, dating back to [1, 7]. Our algorithm will differ from previous approaches in that it works for a wide class of graph clustering objectives, and that all three phases of the algorithm can be specialized for each graph clustering objective. Below, we describe each phase of the algorithm.

We assume that we are given an input graph, denoted by $G_0 = (\mathcal{V}_0, \mathcal{E}_0, A_0)$, as well as the number of partitions requested, k. See Figure 1 for a graphical overview of the multilevel approach.

3.1 Coarsening Phase

Given the initial graph G_0, the graph is repeatedly transformed into smaller and smaller graphs G_1, G_2, ..., G_m such that $|\mathcal{V}_0| > |\mathcal{V}_1| > ... > |\mathcal{V}_m|$. To coarsen a graph from G_i to G_{i+1}, a number of different techniques may be used. In general, sets of nodes in G_i are combined to form supernodes in G_{i+1}. When combining a set of nodes into supernodes, the edge weights between two supernodes in G_{i+1} are taken to be the sum of the edge weights between the nodes in G_i comprising the supernodes.

Our coarsening approach works as follows: given a graph, start with all nodes unmarked. Visit each vertex in a random order. For each unmarked vertex x, if all neighbors of x have been marked, mark x and proceed to the next unmarked vertex. On the other hand, if x has an unmarked neighbor, then merge x with the unmarked vertex y that maximizes

$$\frac{e(x,y)}{w(x)} + \frac{e(x,y)}{w(y)}, \tag{1}$$

where $e(x,y)$ corresponds to the edge weight between vertices x and y and $w(x)$ is the weight of vertex x. Then mark both x and y. Once all vertices are marked, the coarsening for this level is completed.

As we will see when discussing the connection between graph clustering and weighted kernel k-means, different graph clustering objectives induce different vertex weights. For example, in normalized cut, the weight of a vertex is its degree, and (1) reduces to the normalized cut between x and y. For ratio association, (1) simplifies to the heaviest edge criterion of METIS, since all vertices have unit weight.

3.2 Initial Clustering Phase

Eventually, the graph is coarsened to the point where very few nodes remain in the graph. We specify a parameter indicating how small we want the coarsest graph to be; in our experiments, we stop coarsening when the graph has less than $20k$ nodes, where k is the number of desired clusters. At this point, we perform an initial partitioning by clustering the coarsest graph.

One way to obtain an initial partitioning is to use the region growing algorithm of METIS [8]. However, this approach is inadequate for our purposes since it generates clusters of equal size. We have found the best method for initialization to be a spectral algorithm. We use the spectral algorithm of Yu and Shi [10], which we generalize to work with arbitrary weights [4]. Thus our initial clustering is "customized" for different graph clustering objectives. Since the coarsest graph is significantly smaller than the input graph, spectral methods are adequate in terms of speed. We find the spectral methods to yield the best quality of results, though they are somewhat less efficient than the region growing algorithm. Hence, when efficiency is a concern, the region growing algorithm may be used instead.

3.3 Refinement Phase

The final phase of the algorithm is the refinement phase. Given a graph G_i, we form the graph G_{i-1} (G_{i-1} is the same graph used in level $i-1$ in the coarsening phase). We extend the clustering from G_i to G_{i-1} as follows: if a supernode is in a cluster c, then all nodes in G_i formed from that supernode are in cluster c. This yields an initial clustering for the graph, which is then improved using a refinement algorithm. Note that we also run the refinement algorithm on the coarsest graph.

The algorithm terminates after the refinement algorithm is run on the original graph G_0. Since we have a good initial clustering at each level, the refinement algorithm often converges very quickly. Thus, this approach is extremely efficient.

At each refinement step, our algorithm uses an approach inspired by the recently-shown theoretical connection between kernel k-means and spectral clustering [4]. In [4], we proved that each of the graph clustering objectives given earlier may be expressed as special cases of the weighted kernel k-means objective with the correct choice of weights and kernel matrix. Hence, the weighted kernel k-means algorithm can be directly used to locally optimize these graph clustering objectives. Each iteration of this algorithm costs $O(nz)$ time, where nz is the number of nonzero entries in the kernel matrix.

Algorithm 1 describes the weighted kernel k-means algorithm. The input is the kernel matrix K, the number of clusters k, weights for the points w, an optional maximum number of iterations, and an optional initial clustering, denoted by $\{\pi_c^{(0)}\}_{c=1}^k$.

For the Kernighan-Lin (K-L) objective, which requires equally sized clusters, an incremental kernel k-means algorithm (in which points are swapped to improve the objective function value) can be used to locally optimize the objective.

In Table 2, we display each of the earlier graph clustering objectives, along with the choice of weights and kernel required for the weighted kernel k-means algorithm to monotonically optimize the given graph clustering objective. The matrix D is a diagonal matrix whose entries correspond to the sum of the rows of A, the graph affinity matrix. Finally, σ is a real number chosen to be large enough that K is positive definite. As long as K is positive definite, we

Algorithm 1: Refinement Algorithm.

Weighted_Kernel_kmeans(K, k, w, t_{max}, $\{\pi_c^{(0)}\}_{c=1}^k$, $\{\pi_c\}_{c=1}^k$)

Input: K: kernel matrix, k: number of clusters, w: weights for each point, t_{max}: optional maximum number of iterations, $\{\pi_c^{(0)}\}_{c=1}^k$: optional initial clustering

Output: $\{\pi_c\}_{c=1}^k$: final clustering of the points

1. If no initial clustering is given, initialize the k clusters $\pi_1^{(0)}, ..., \pi_k^{(0)}$ randomly. Set $t = 0$.
2. For each row i of K and every cluster c, compute

$$d(i, \mathbf{m}_c) = K_{ii} - \frac{2\sum_{j\in\pi_c^{(t)}} w_j K_{ij}}{\sum_{j\in\pi_c^{(t)}} w_j} + \frac{\sum_{j,l\in\pi_c^{(t)}} w_j w_l K_{jl}}{(\sum_{j\in\pi_c^{(t)}} w_j)^2}.$$

3. Find $c^*(i) = \text{argmin}_c d(i, \mathbf{m}_c)$, resolving ties arbitrarily. Compute the updated clusters as

$$\pi_c^{(t+1)} = \{i \ : \ c^*(i) = c\}.$$

4. If not converged or $t_{max} > t$, set $t = t+1$ and go to Step 3; Otherwise, stop and output final clusters $\{\pi_c^{(t+1)}\}_{c=1}^k$.

Objective	Node Weights	Kernel Matrix
Ratio Association	1 $\forall$ nodes	$K = \sigma I + A$
Ratio Cut	1 $\forall$ nodes	$K = \sigma I - D + A$
K-L Objective	1 $\forall$ nodes	$K = \sigma I - D + A$
Normalized Cut	Deg. of node	$K = \sigma D^{-1} + D^{-1}AD^{-1}$

Table 2: Popular graph clustering objectives with corresponding weights and kernels given the affinity matrix A

Graph name	#nodes	#edges	Description
DATA	2851	15093	finite element mesh
3ELT	4720	13722	finite element mesh
UK	4824	6837	finite element mesh
ADD32	4960	9462	32-bit adder
WHITAKER3	9800	28989	finite element mesh
CRACK	10240	30380	finite element mesh
FE_4ELT2	11143	32818	finite element mesh
MEMPLUS	17758	54196	memory circuit
BCSSTK30	28294	1007284	stiffness matrix

Table 3: Test graphs

can theoretically guarantee that the algorithm monotonically optimizes the corresponding graph clustering objective. Initialization can be done randomly, or by using another clustering algorithm, such as METIS or spectral clustering. In the case of our multilevel algorithm, initialization is obtained from the clustering from the previous level, or the initial clustering in the case of the coarsest graph.

An extension to the basic batch algorithm presented in [4] uses local search to avoid local optima. To avoid this problem, it has been shown [5] that local search can significantly improve results by allowing the algorithm to escape such poor optima. The local search algorithm considers the effect on the objective function of moving a point from one cluster to another. It chooses a chain of moves that causes the greatest improvement in the objective function value. We incorporate this local search procedure to improve the quality of our weighted kernel k-means batch refinement scheme.

4. EXPERIMENTS

In this section, we present a number of experiments to show that our multilevel weighted kernel k-means algorithm outperforms spectral methods in terms of quality (normalized cut and ratio association values) as well as computation time. We also present results of clustering the 1.2 million node IMDB movie data set graph. In our experiments, we use the spectral algorithm from [10], generalized to work for both normalized cut and ratio association. This algorithm is used as the benchmark spectral algorithm in our experiments, as well as the method for clustering the coarsest graph in our multilevel implementation. To speed up computation, we considered only "boundary points" when running weighted kernel k-means; further discussion is omitted due to lack of space. All the experiments are performed on a Linux machine with 2.4 GHz Pentium IV processor and 1 GB main memory.

Additionally, we use the following acronyms: mlkkm(0) stands for our multilevel kernel k-means algorithm with no local search, mlkkm(20) is our multilevel algorithm with a local search chain length of 20, kkm stands for kernel k-means with random initialization, NCV is short for normalized cut value, and RAV is short for ratio association value.

4.1 Clustering of Benchmark Graphs

Table 3 lists 9 test graphs[1] from various sources and different application domains. Some of them are benchmark matrices used to test METIS.

	mlkkm(0)	kkm	METIS
NCV	**2308**	4788	2643
RAV	**18526**	1349	12744

Table 4: Normalized cut values (NCV) and ratio association values (RAV) for obtaining 5000 clusters of the IMDB movie data set using our multilevel algorithm (mlkkm(0)), kernel k-means (kkm) and METIS

On each of the benchmark graphs, we compared our multilevel algorithms, mlkkm(0) and mlkkm(20), with the spectral algorithm in terms of objective function values and computation time. We report results for both ratio association and normalized cut, and when 64 clusters are computed. We do not report comparisons between our multilevel algorithm and METIS; in [4], we showed that the spectral algorithm is superior to METIS in terms of better ratio association and normalized cut values.

Figure 2 shows the relative performance of the spectral algorithm compared with our multilevel algorithm in terms of RAV and computation time. In the left panel, for each graph we plot the ratio of the RAV of all methods to that of mlkkm(20) — ratios less than 1 indicate that mlkkm(20) produces higher RAVs (and thus performs better). We can see that 100% of the RAVs produced using mlkkm(20) are higher than those obtained by the spectral algorithm. Also, we see that mlkkm(20) performs consistently better than mlkkm(0), which implies that local search improves the RAV in all cases (maximally by 3%).

In the right panel of Figure 2, we plot the computation time for the three methods. It is clear that our multilevel algorithm is much faster than the spectral algorithm. For example, in the case of ADD32, the spectral algorithm is 48 times slower than mlkkm(20) and 55 times slower than mlkkm(0).

Results for the normalized cut objective appear in Figure 3. For each graph, we plot the ratio of NCV of other methods to that of mlkkm(20). As opposed to ratio association, values that are larger than 1 in Figure 3 indicate that mlkkm(20) produces lower NCVs (and thus performs better). We see that 67% of the NCVs produced using mlkkm(20) are lower than those produced using the spectral algorithm. We also see that mlkkm(20), which utilizes the local search technique, performs much better than mlkkm(0) in many cases.

In terms of computation time, our multilevel methods again outperform the spectral algorithm in all cases. For ADD32, spectral is 58 times slower than mlkkm(20) and 60 times slower than mlkkm(0).

4.2 The IMDB Movie Data Set — A Case Study

The IMDB data set[2] contains information about movies, music bands, actors, movie festivals and events. By connecting actors and movies or events in which they participate, we form a sparse undirected graph with approximately 1.2 million nodes and 7.6 million edges.

It is impractical to run a spectral algorithm directly on this data set to produce 5000 clusters not only because com-

[1]Downloaded from http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/

[2]Downloaded from http://www.imdb.com/interfaces

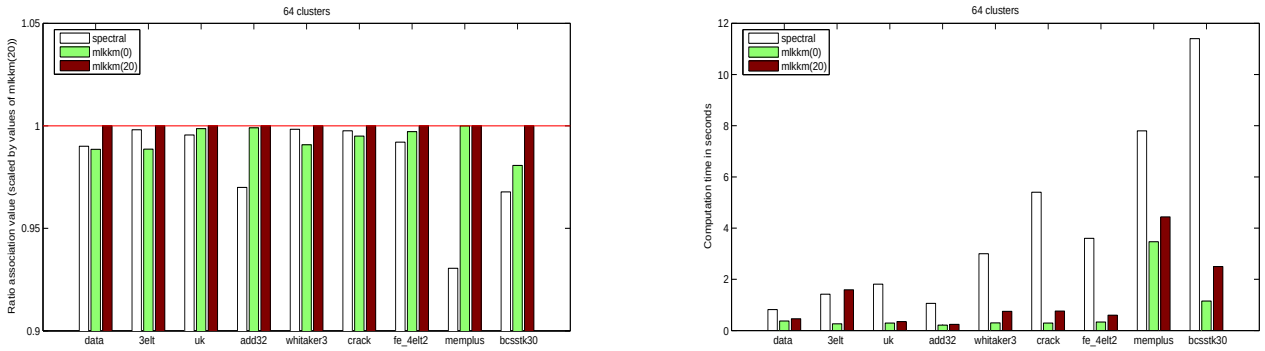

Figure 2: Quality and computation time of our multilevel methods compared with the benchmark spectral algorithm. The left panel plots the ratio of the ratio association value of every algorithm to that of mlkkm(20) for 64 clusters. Note that bars below the baseline correspond to cases where mlkkm(20) performs better. The right panel compares computation times for generating 64 clusters using different algorithms. As an example, on the ADD32 graph, the spectral algorithm is 48 times slower than mlkkm(20) and 58 times slower than mlkkm(0).

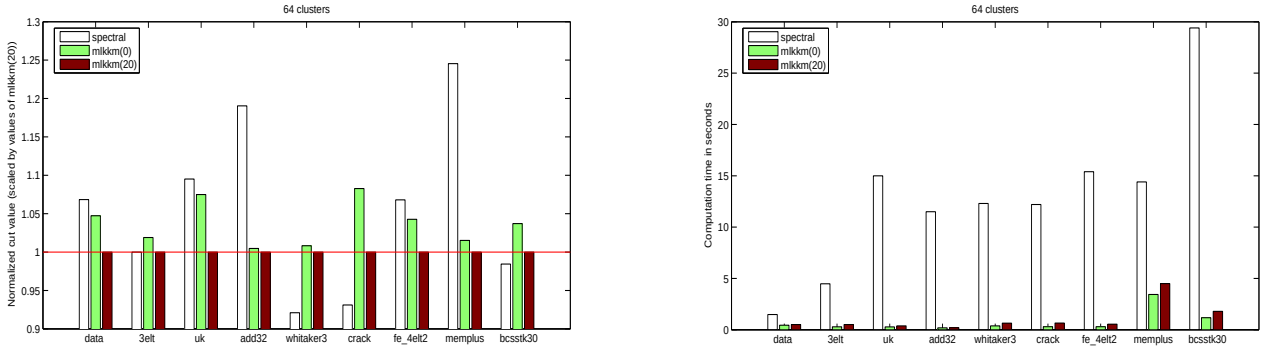

Figure 3: Quality and computation time of our multilevel methods compared with the benchmark spectral algorithm. The left panel plots the ratio of the normalized cut value of every algorithm to that of mlkkm(20) for 64 clusters. Note that bars above the baseline correspond to the cases where mlkkm(20) performs better. The right panel compares computation times for generating 64 clusters. As an example, on the ADD32 graph, the spectral algorithm is 58 times slower than mlkkm(20) and 60 times slower than mlkkm(0).

Movies	Actors
Harry Potter and the Sorcerer's Stone (2001)	Daniel Radcliffe, Rupert Grint, Emma Watson, Tom Felton
Harry Potter and the Chamber of Secrets (2002)	Peter Best, Sean Biggerstaff, Scott Fern, Alfred Enoch, Joshua Herdman
Harry Potter and the Prisoner of Azkaban (2004)	Harry Melling, Matthew Lewis, Devon Murray, Robert Pattinson
Harry Potter and the Goblet of Fire (2005)	James Phelps, Oliver Phelps, Edward Randell, Jamie Waylett
Harry Potter: Behind the Magic (2001 TV)	Shefali Chowdhury, Katie Leung, Bonnie Wright, Stanislav Ianevski
Harry Potter und die Kammer des Schreckens: Das grobe RTL Special zum Film (2002 TV)	Jamie Yeates, Chris Rankin

Table 5: A Selection of Movies and Actors in Cluster 633

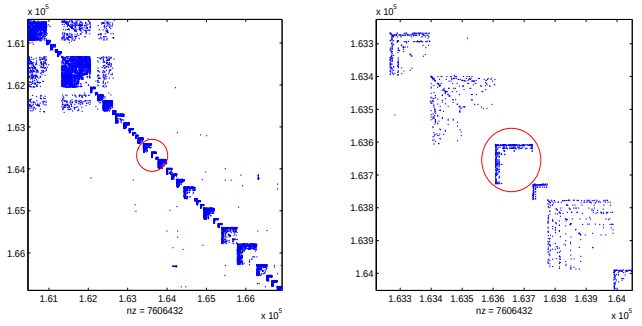

Figure 4: Graphical Plots of Cluster 633 of IMDB (Harry Potter). The right plot shows a closer view of the cluster circled in the left plot.

puting 5000 eigenvectors of a graph of this size would be extremely slow but also because storing 5000 eigenvectors in main memory requires 24 GB, which is prohibitive. Thus our multilevel algorithm is also considerably more efficient in terms of memory usage in addition to computation time. Since we cannot run spectral methods on this data set, we compare our multilevel algorithm (mlkkm(0)) to METIS and weighted kernel k-means with random initialization (kkm). It takes our multilevel algorithm approximately twenty-five minutes to cluster this graph into 5000 clusters, and the cluster sizes range from 13 to 7616 (for the normalized cut objective). Table 4 shows the normalized cut values and ratio association values for 5000 clusters generated using these three methods. We see that our multilevel algorithm is markedly superior to kkm: its NCV is twice as high as the NCV of mlkkm(0) and the RAV of kkm is less than one tenth of the RAV of mlkkm(0). Comparing METIS and mlkkm(0), the latter is superior and achieves a RAV that is 50% higher and a NCV that is 10% lower.

To demonstrate the quality of results, we discuss a sampling of the produced clusters. After clustering, we rearrange the rows and columns such that rows (columns) in the same cluster are adjacent to one another. Cluster 633, circled in the right plot of Figure 4, is of size 121 and mainly contains "Harry Potter" movies and the actors in these movies. The left column of Table 5 lists movies in the cluster, where we see 3 Harry Potter movies that were released in the past and 1 to be released in November, 2005. There are also 2 other documentary TV programs. The right column of Table 5 lists a selection of some of the actors in the cluster, where we see the major cast members of the Harry Potter movies, such as Daniel Radcliffe, Rupert Grint, Emma Watson, etc.

Cluster 3537 contains the short series "Festival número" (No. 1-12), shot in year 1965; cluster 4400 contains the Korean movie "Chunhyang" and 19 of its cast members who acted only in this movie in the database. Other small clusters follow this pattern, i.e., most of them are about one movie or movie series and contain cast members that acted only in this movie or movie series. Popular actors, directors or well-known movie festivals are associated with more people, so generally they belong to much larger clusters.

5. CONCLUSIONS

We have presented a new multilevel algorithm for graph clustering. This algorithm has key benefits over existing graph clustering algorithms. Unlike previous multilevel algorithms such as METIS, our algorithm is general and captures a number of graph clustering objectives. This frees us from the restriction of equal-size clusters. It has advantages over spectral methods in that the multilevel approach is considerably faster and gives better final objective function values.

Using a number of benchmark test graphs, we demonstrated that, in general, our algorithm produces better ratio association values and normalized cut values when using spectral initialization at the coarsest level. As the coarsest graph is significantly smaller than the input graph, spectral methods are feasible in this case, and the running time of the algorithm is still much faster than spectral clustering on the input graph. Spectral methods have attracted intense study over the last several years as a powerful method for graph clustering; our results indicate that this may not be the most powerful technique for partitioning graphs. Furthermore, given an input pairwise similarity matrix between data vectors, our approach can be viewed as a fast multilevel algorithm for optimizing the kernel k-means objective.

Our approach also consumes far less memory than spectral methods. Spectral clustering of a 1.2 million node movie network into 5000 clusters would require several gigabytes of storage. On the other hand, we were able to cluster this network in approximately twenty-five minutes without any significant memory overhead.

Acknowledgments. This research was supported by NSF grant CCF-0431257, NSF Career Award ACI-0093404, and NSF-ITR award IIS-0325116.

6. REFERENCES

[1] T. Bui and C. Jones. A heuristic for reducing fill-in in sparse matrix factorization. In *6th SIAM Conference on Parallel Processing for Scientific Computing*, pages 445–452, 1993.

[2] P. Chan, M. Schlag, and J. Zien. Spectral k-way ratio cut partitioning. *IEEE Trans. CAD-Integrated Circuits and Systems*, 13:1088–1096, 1994.

[3] I. Dhillon, Y. Guan, and B. Kulis. Kernel k-means, spectral clustering and normalized cuts. In *Proc. 10th ACM KDD Conference*, 2004.

[4] I. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical Report TR-04-25, University of Texas at Austin, 2004.

[5] I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proceedings of The 2002 IEEE International Conference on Data Mining*, pages 131–138, 2002.

[6] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM J. Res. Development*, 17:422–425, 1973.

[7] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories, 1993.

[8] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1999.

[9] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[10] S. X. Yu and J. Shi. Multiclass spectral clustering. In *International Conference on Computer Vision*, 2003.