Weighted Graph Cuts without Eigenvectors: A Multilevel Approach

Inderjit S. Dhillon, *Member, IEEE*, Yuqiang Guan and Brian Kulis

***Abstract*— A variety of clustering algorithms have recently been proposed to handle data that is not linearly separable; spectral clustering and kernel *k*-means are two of the main methods. In this paper, we discuss an equivalence between the objective functions used in these seemingly different methods—in particular, a general weighted kernel *k*-means objective is *mathematically equivalent* to a weighted graph clustering objective. We exploit this equivalence to develop a fast, high-quality multilevel algorithm that directly optimizes various weighted graph clustering objectives, such as the popular ratio cut, normalized cut, and ratio association criteria. This eliminates the need for any eigenvector computation for graph clustering problems, which can be prohibitive for very large graphs. Previous multilevel graph partitioning methods, such as Metis, have suffered from the restriction of equal-sized clusters; our multilevel algorithm removes this restriction by using kernel k-means to optimize weighted graph cuts. Experimental results show that our multilevel algorithm outperforms a state-of-the-art spectral clustering algorithm in terms of speed, memory usage, and quality. We demonstrate that our algorithm is applicable to large-scale clustering tasks such as image segmentation, social network analysis and gene network analysis.**

***Index Terms*— Clustering, Data Mining, Segmentation, Kernel k-means, Spectral Clustering, Graph Partitioning**

I. Introduction

CLUSTERING is an important problem with many applications, and a number of different algorithms and methods have emerged over the years. In this paper, we discuss an equivalence between two seemingly different methods for clustering non-linearly separable data: kernel *k*-means and spectral clustering. Using this equivalence, we design a fast kernel-based multilevel graph clustering algorithm that outperforms spectral clustering methods in terms of speed, memory usage and quality.

The kernel *k*-means algorithm [1] is a generalization of the standard *k*-means algorithm [2]. By implicitly mapping data to a higher-dimensional space, kernel *k*-means can discover clusters that are non-linearly separable in input space. This provides a major advantage over standard *k*-means, and allows data clustering given a positive definite matrix of similarity values.

On the other hand, graph clustering (also called graph partitioning) algorithms focus on clustering nodes of a graph [3], [4]. Spectral methods have been used effectively for solving a number of graph clustering objectives, including ratio cut [5] and normalized cut [6]. Such an approach has been useful in many areas, such as circuit layout [5] and image segmentation [6]. Recently, spectral clustering, so-called because of the usage of eigenvectors, has enjoyed immense popularity in various machine learning tasks [7].

In this paper, we generalize and extend recent results [8], [9], [10], [11] on connections between vector-based and graph-based clustering to provide a unifying mathematical connection between kernel k-means and graph clustering objectives. In particular, a weighted form of the kernel *k*-means objective is seen to be mathematically equivalent to a general, weighted graph clustering objective. Such an equivalence has an immediate implication: we may use the weighted kernel *k*-means algorithm to locally optimize a number of graph clustering objectives, and conversely, spectral methods may be employed for weighted kernel *k*-means. In cases where eigenvector computation is prohibitive (for example, if many eigenvectors of a very large matrix are required), the weighted kernel *k*-means algorithm may be more desirable than spectral methods. Previous work in this area has not considered such a kernel-based approach, nor general weighted graph clustering objectives.

The benefits of using kernels in our analysis is highlighted in our development of a new multilevel graph clustering algorithm that can be specialized to a wide class of graph clustering objectives. In multilevel algorithms, the input graph is repeatedly coarsened level by level until only a small number of nodes remain. An initial clustering is performed on the coarsened graph, and then this clustering is refined as the graph is uncoarsened level by level. These methods are extremely fast and give high-quality partitions. However, earlier multilevel methods, such as Metis [12] and Chaco [13], force clusters to be of nearly equal size, and are all based on optimizing the Kernighan-Lin objective [14]. In contrast, our multilevel algorithm removes the restriction of equal cluster size, and uses the weighted kernel k-means algorithm during the refinement phase to directly optimize various weighted graph cuts. Furthermore, the multilevel algorithm only requires memory on the order of the input graph; on the other hand, spectral methods that compute k eigenvectors require $O(nk)$ storage, where n is the number of data points and k is the number of clusters. This makes the multilevel algorithm scalable to much larger data sets than standard spectral methods.

We present experimental results on a number of interesting applications, including gene network analysis, social network analysis and image segmentation. We also compare variants of our multilevel algorithm to a state-of-the-art spectral method. These variants outperform or are competitive with the spectral clustering algorithms in terms of optimizing the corresponding objective functions. Furthermore, all variants of our algorithm are significantly faster than spectral methods and require far less memory usage. In fact, the spectral method cannot be applied to our social network example due to the size of the data set.

A word about our notation. Capital letters such as A, X and Φ represent matrices, while lower-case bold letters such as $\mathbf{a}$ and $\mathbf{b}$ represent vectors. Script letters such as $\mathcal{V}$ and $\mathcal{E}$ represent sets.

Inderjit S. Dhillon and Brian Kulis are with the Department of Computer Sciences, University of Texas at Austin, 1 University Station, Austin, TX, 78712, USA.

Yuqiang Guan is with Microsoft Corporation, 1 Microsoft Way, Redmond, WA, 98052, USA.

Polynomial Kernel	$\kappa(\mathbf{a}_i, \mathbf{a}_j) = (\mathbf{a}_i \cdot \mathbf{a}_j + c)^d$
Gaussian Kernel	$\kappa(\mathbf{a}_i, \mathbf{a}_j) = \exp(-\|\mathbf{a}_i - \mathbf{a}_j\|^2/2\alpha^2)$
Sigmoid Kernel	$\kappa(\mathbf{a}_i, \mathbf{a}_j) = \tanh(c(\mathbf{a}_i \cdot \mathbf{a}_j) + \theta)$

TABLE I
EXAMPLES OF POPULAR KERNEL FUNCTIONS

We use $\|\mathbf{a}\|$ to represent the L^2 norm of a vector $\mathbf{a}$, and $\|X\|_F$ for the Frobenius norm of a matrix X. Finally, $\mathbf{a} \cdot \mathbf{b}$ is the inner product between the vectors $\mathbf{a}$ and $\mathbf{b}$.

II. KERNEL K-MEANS

Given a set of vectors $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n$, the *k*-means algorithm seeks to find clusters $\pi_1, \pi_2, ...\pi_k$ that minimize the objective function:

$$\mathcal{D}(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{\mathbf{a}_i \in \pi_c} \|\mathbf{a}_i - \mathbf{m}_c\|^2, \text{ where } \mathbf{m}_c = \frac{\sum_{\mathbf{a}_i \in \pi_c} \mathbf{a}_i}{|\pi_c|}.$$

Note that the c-th cluster is denoted by π_c, a clustering or partitioning by $\{\pi_c\}_{c=1}^k$, while the centroid or mean of cluster π_c is denoted by $\mathbf{m}_c$.

A disadvantage to standard *k*-means is that clusters can only be separated by a hyperplane; this follows from the fact that squared Euclidean distance is used as the distortion measure. To allow non-linear separators, kernel *k*-means [1] first uses a function ϕ to map data points to a higher-dimensional feature space, and then applies *k*-means in this feature space. Linear separators in the feature space correspond to nonlinear separators in the input space.

The kernel *k*-means objective can be written as a minimization of:

$$\begin{aligned} \mathcal{D}(\{\pi_c\}_{c=1}^k) &= \sum_{c=1}^{k} \sum_{\mathbf{a}_i \in \pi_c} \|\phi(\mathbf{a}_i) - \mathbf{m}_c\|^2, \\ \text{where } \mathbf{m}_c &= \frac{\sum_{\mathbf{a}_i \in \pi_c} \phi(\mathbf{a}_i)}{|\pi_c|}. \end{aligned}$$

The squared distance $\|\phi(\mathbf{a}_i) - \mathbf{m}_c\|^2$ may be rewritten as:

$$\phi(\mathbf{a}_i) \cdot \phi(\mathbf{a}_i) - \frac{2\sum_{\mathbf{a}_j \in \pi_c} \phi(\mathbf{a}_i) \cdot \phi(\mathbf{a}_j)}{|\pi_c|} + \frac{\sum_{\mathbf{a}_j, \mathbf{a}_l \in \pi_c} \phi(\mathbf{a}_j) \cdot \phi(\mathbf{a}_l)}{|\pi_c|^2}.$$

Thus, only inner products are used in this computation. As a result, given a *kernel* matrix K, where $K_{ij} = \phi(\mathbf{a}_i) \cdot \phi(\mathbf{a}_j)$, distances between points and centroids can be computed without knowing explicit representations of $\phi(\mathbf{a}_i)$ and $\phi(\mathbf{a}_j)$. It can be shown that *any* positive semidefinite matrix K can be thought of as a kernel matrix [15].

A kernel function κ is commonly used to map the original points to inner products. See Table I for commonly used kernel functions; $\kappa(\mathbf{a}_i, \mathbf{a}_j) = K_{ij}$.

A. Weighted kernel k-means

We now introduce a weighted version of the kernel *k*-means objective function, first described in [9]. As we will see later, the weights play a crucial role in showing an equivalence to graph clustering. The weighted kernel *k*-means objective function is expressed as:

$$\begin{aligned} \mathcal{D}(\{\pi_c\}_{c=1}^k) &= \sum_{c=1}^{k} \sum_{\mathbf{a}_i \in \pi_c} w_i \|\phi(\mathbf{a}_i) - \mathbf{m}_c\|^2, \\ \text{where } \mathbf{m}_c &= \frac{\sum_{\mathbf{a}_i \in \pi_c} w_i \phi(\mathbf{a}_i)}{\sum_{\mathbf{a}_i \in \pi_c} w_i}, \end{aligned}$$

and the weights w_i are non-negative. Note that $\mathbf{m}_c$ represents the "best" cluster representative since

$$\mathbf{m}_c = \text{argmin}_{\mathbf{z}} \sum_{\mathbf{a}_i \in \pi_c} w_i \|\phi(\mathbf{a}_i) - \mathbf{z}\|^2.$$

As before, we compute distances only using inner products, since $\|\phi(\mathbf{a}_i) - \mathbf{m}_c\|^2$ equals

$$\begin{aligned} \phi(\mathbf{a}_i) \cdot \phi(\mathbf{a}_i) - \frac{2\sum_{\mathbf{a}_j \in \pi_c} w_j \phi(\mathbf{a}_i) \cdot \phi(\mathbf{a}_j)}{\sum_{\mathbf{a}_j \in \pi_c} w_j} \\ + \frac{\sum_{\mathbf{a}_j, \mathbf{a}_l \in \pi_c} w_j w_l \phi(\mathbf{a}_j) \cdot \phi(\mathbf{a}_l)}{(\sum_{\mathbf{a}_j \in \pi_c} w_j)^2}. \end{aligned} \quad (1)$$

Using the kernel matrix K, the above may be rewritten as:

$$K_{ii} - \frac{2\sum_{\mathbf{a}_j \in \pi_c} w_j K_{ij}}{\sum_{\mathbf{a}_j \in \pi_c} w_j} + \frac{\sum_{\mathbf{a}_j, \mathbf{a}_l \in \pi_c} w_j w_l K_{jl}}{(\sum_{\mathbf{a}_j \in \pi_c} w_j)^2}. \quad (2)$$

ALGORITHM 1: Basic Batch Weighted Kernel k-means.

KERNEL_KMEANS_BATCH(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 clusters
Output: $\{\pi_c\}_{c=1}^k$: final clustering of the points
1. If no initial clustering is given, initialize the k clusters $\pi_1^{(0)}, ..., \pi_k^{(0)}$ (e.g., randomly). Set $t = 0$.
2. For each point $\mathbf{a}_i$ and every cluster c, compute

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

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

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

4. Set $t = t + 1$. If not converged or $t < t_{max}$, go to Step 2; otherwise, stop and output final clusters $\{\pi_c^{(t)}\}_{c=1}^k$.

B. Computational Complexity

We now analyze the computational complexity of a basic weighted kernel *k*-means algorithm presented in Algorithm 1. The algorithm is a direct generalization of standard *k*-means. As in standard *k*-means, given the current centroids, the closest centroid for each point is computed. After this, the clustering is re-computed. These two steps are repeated until the change in the objective function value is sufficiently small. It can be shown that this algorithm monotonically converges as long as K is positive semi-definite so that it can be interpreted as a Gram matrix.

It is clear that the bottleneck in the kernel *k*-means algorithm is Step 2, i.e., the computation of distances $d(\mathbf{a}_i, \mathbf{m}_c)$. The first term, K_{ii}, is a constant for $\mathbf{a}_i$ and does not affect the assignment of $\mathbf{a}_i$ to clusters. The second term requires $O(n)$ computation for every

data point, leading to a cost of $O(n^2)$ per iteration. The third term is fixed for cluster c, so in each iteration it can be computed once and stored; over all clusters, this takes $O(\sum_c |\pi_c|^2) = O(n^2)$ operations. Thus the complexity is $O(n^2)$ scalar operations per iteration. However, for a sparse matrix K, each iteration can be adapted to cost $O(nz)$ operations, where nz is the number of non-zero entries in the matrix ($nz = n^2$ for a dense kernel matrix). If we are using a kernel function κ to generate the kernel matrix from our data, computing K usually requires $O(n^2 m)$ operations, where m is the original data dimension. If the total number of iterations is τ, then the time complexity of Algorithm 1 is $O(n^2(\tau + m))$ if we are given data vectors as input, or $O(nz \cdot \tau)$ if we are given a positive definite matrix as input.

III. Graph Clustering

We now shift our focus to a different approach to clustering data, namely graph clustering. In this model of clustering, we are given a graph $G = (\mathcal{V}, \mathcal{E}, A)$, which is made up 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 adjacency matrix A is $|\mathcal{V}| \times |\mathcal{V}|$ whose non-zero entries equal the edge weights (an entry of A is 0 if there is no edge between the corresponding vertices).

Let us denote $\text{links}(\mathcal{A}, \mathcal{B})$ to be the sum of the edge weights between nodes in $\mathcal{A}$ and $\mathcal{B}$. In other words,

$$\text{links}(\mathcal{A}, \mathcal{B}) = \sum_{i \in \mathcal{A}, j \in \mathcal{B}} A_{ij}.$$

Furthermore, let the degree of $\mathcal{A}$ be the links of nodes in $\mathcal{A}$ to all the vertices, i.e. $\text{degree}(\mathcal{A}) = \text{links}(\mathcal{A}, \mathcal{V})$.

The graph clustering problem seeks to partition the graph into k disjoint partitions or clusters $\mathcal{V}_1, ..., \mathcal{V}_k$ such that their union is $\mathcal{V}$. A number of different graph clustering objectives have been proposed and studied, and we will focus on the most prominent ones.

Ratio Association. The ratio association (also called average association) objective [6] aims to maximize within-cluster association relative to the size of the cluster. The objective can be written as

$$RAssoc(G) = \max_{\mathcal{V}_1,...,\mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V}_c)}{|\mathcal{V}_c|}.$$

Ratio Cut. The ratio cut objective [5] differs from ratio association in that it seeks to minimize the cut between clusters and the remaining vertices. It is expressed as

$$RCut(G) = \min_{\mathcal{V}_1,...,\mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{|\mathcal{V}_c|}.$$

Kernighan-Lin Objective. This objective [14] is nearly identical to the ratio cut objective, except that the partitions are required to be of equal size. Although this objective is generally presented for $k = 2$ clusters, we can easily generalize it for arbitrary k. For simplicity, we assume that the number of vertices $|\mathcal{V}|$ is divisible by k. Then, we write the objective as

$$\begin{aligned} KLObj(G) &= \min_{\mathcal{V}_1,...,\mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{|\mathcal{V}_c|}, \\ \text{subject to } |\mathcal{V}_c| &= |\mathcal{V}|/k \quad \forall c = 1, ..., k. \end{aligned}$$

Normalized Cut. The normalized cut objective [6], [16] is one of the most popular graph clustering objectives and seeks to minimize the cut relative to the degree of a cluster instead of its size. The objective is expressed as

$$NCut(G) = \min_{\mathcal{V}_1,...,\mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{\text{degree}(\mathcal{V}_c)}.$$

It should be noted that minimizing the normalized cut is equivalent to maximizing the normalized association [16], since $\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c) = \text{degree}(\mathcal{V}_c) - \text{links}(\mathcal{V}_c, \mathcal{V}_c)$.

General Weighted Graph Cuts/Association. We can generalize the association and cut problems to weighted variants. This will prove useful for building a general connection to weighted kernel *k*-means. We introduce a weight w_i for each node of the graph, and for each cluster $\mathcal{V}_c$, define $w(\mathcal{V}_c) = \sum_{i \in \mathcal{V}_c} w_i$. We generalize the association problem to be:

$$WAssoc(G) = \max_{\mathcal{V}_1,...,\mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V}_c)}{w(\mathcal{V}_c)}.$$

Similarly, for cuts:

$$WCut(G) = \min_{\mathcal{V}_1,...,\mathcal{V}_k} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{w(\mathcal{V}_c)}.$$

Ratio association is a special case of $WAssoc$ where all weights are equal to one (and hence the weight of a cluster is simply the number of vertices in it), and normalized association is a special case where the weight of a node i is equal to its degree (the sum of row i of the adjacency matrix A). An analogous result is true for $WCut$. More generally, weights may be fixed to other values based on the clustering problem. For example, [17] proposed the use of Sinkhorn normalization to obtain vertex weights in order that the normalized adjacency matrix is the closest doubly stochastic matrix to the input in relative entropy.

For the Kernighan-Lin objective, an incremental algorithm is traditionally used to swap chains of vertex pairs; for more information, see [14]. For ratio association, ratio cut, and normalized cut, the algorithms traditionally used to optimize the objectives employ eigenvectors of the adjacency matrix, or a matrix derived from the adjacency matrix. We discuss these spectral solutions in the next section, where we prove that the $WAssoc$ objective is equivalent to the weighted kernel *k*-means objective, and that $WCut$ can be viewed as a special case of $WAssoc$, and thus as a special case of weighted kernel *k*-means.

IV. Equivalence of the Objectives

At first glance, the two approaches to clustering presented in the previous two sections appear to be unrelated. In this section, we first express the weighted kernel *k*-means objective as a trace maximization problem. We then rewrite the weighted graph association and graph cut problems identically as matrix trace maximizations, thus showing that the two objectives are mathematically equivalent. Finally, we discuss the connection to spectral methods, and show how to enforce positive definiteness of the adjacency matrix in order to guarantee convergence of the weighted kernel *k*-means algorithm.

A. Weighted Kernel k-means as Trace Maximization

Let s_c be the sum of the weights in cluster c, i.e. $s_c = \sum_{\mathbf{a}_i \in \pi_c} w_i$. Define the $n \times k$ matrix Z:

$$Z_{ic} = \begin{cases} \frac{1}{s_c^{1/2}} & \text{if } \mathbf{a}_i \in \pi_c, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, the columns of Z are mutually orthogonal, as they capture disjoint cluster memberships. Suppose Φ is the matrix of all the $\phi(\mathbf{a}_i)$ vectors, $i = 1, \ldots, n$, and W is the diagonal matrix of weights. The matrix $\Phi W Z Z^T$ has column i equal to the mean vector of the cluster that contains $\mathbf{a}_i$. Thus, the weighted kernel *k*-means objective may be written as:

$$\begin{aligned} \mathcal{D}(\{\pi_c\}_{c=1}^k) &= \sum_{c=1}^{k} \sum_{\mathbf{a}_i \in \pi_c} w_i \|\phi(\mathbf{a}_i) - \mathbf{m}_c\|^2 \\ &= \sum_{i=1}^{n} w_i \|\Phi_{\cdot i} - (\Phi W Z Z^T)_{\cdot i}\|^2, \end{aligned}$$

where $\Phi_{\cdot i}$ denotes the i-th column of the matrix Φ. Let $\tilde{Y} = W^{1/2} Z$; observe that $\tilde{Y}$ is an orthonormal matrix ($\tilde{Y}^T \tilde{Y} = I_k$). Thus:

$$\begin{aligned} \mathcal{D}(\{\pi_c\}_{c=1}^k) &= \sum_{i=1}^{n} w_i \|\Phi_{\cdot i} - (\Phi W^{1/2} \tilde{Y} \tilde{Y}^T W^{-1/2})_{\cdot i}\|^2 \\ &= \sum_{i=1}^{n} \|\Phi_{\cdot i} w_i^{1/2} - (\Phi W^{1/2} \tilde{Y} \tilde{Y}^T)_{\cdot i}\|^2 \\ &= \|\Phi W^{1/2} - \Phi W^{1/2} \tilde{Y} \tilde{Y}^T\|_F^2. \end{aligned}$$

Since trace(AB) = trace(BA), trace($A^T A$) = $\|A\|_F^2$ and trace($A+B$) = trace(A) + trace(B), we can rewrite $\mathcal{D}(\{\pi_c\}_{c=1}^k)$

$$\begin{aligned} &= \text{trace}(W^{1/2}\Phi^T\Phi W^{1/2} - W^{1/2}\Phi^T\Phi W^{1/2}\tilde{Y}\tilde{Y}^T \\ &\quad -\tilde{Y}\tilde{Y}^T W^{1/2}\Phi^T\Phi W^{1/2} + \tilde{Y}\tilde{Y}^T W^{1/2}\Phi^T\Phi W^{1/2}\tilde{Y}\tilde{Y}^T) \\ &= \text{trace}(W^{1/2}\Phi^T\Phi W^{1/2}) - \text{trace}(\tilde{Y}^T W^{1/2}\Phi^T\Phi W^{1/2}\tilde{Y}). \end{aligned}$$

We note that the kernel matrix K is equal to $\Phi^T\Phi$, and that trace($W^{1/2}KW^{1/2}$) is a constant. Hence, the minimization of the weighted kernel *k*-means objective function is equivalent to:

$$\max_{\tilde{Y}} \ \text{trace}(\tilde{Y}^T W^{1/2} K W^{1/2} \tilde{Y}), \tag{3}$$

where $\tilde{Y}$ is the orthonormal $n \times k$ matrix that is proportional to the square root of the weight matrix W as detailed above.

B. Graph Clustering as Trace Maximization

With this derivation complete, we can now show how each of the graph clustering objectives may be written as trace maximizations as well.

Ratio Association. The simplest objective to transform into a trace maximization is ratio association. Let us introduce an indicator vector $\mathbf{x}_c$ for partition c, i.e. $\mathbf{x}_c(i) = 1$ if cluster c contains vertex i. Then the ratio association objective equals

$$\max \left\{ \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V}_c)}{|\mathcal{V}_c|} = \sum_{c=1}^{k} \frac{\mathbf{x}_c^T A \mathbf{x}_c}{\mathbf{x}_c^T \mathbf{x}_c} = \sum_{c=1}^{k} \tilde{\mathbf{x}}_c^T A \tilde{\mathbf{x}}_c \right\},$$

where A is the graph adjacency matrix and $\tilde{\mathbf{x}}_c = \mathbf{x}_c / (\mathbf{x}_c^T \mathbf{x}_c)^{1/2}$. The equalities hold since $\mathbf{x}_c^T \mathbf{x}_c$ gives us the size of partition c, while $\mathbf{x}_c^T A \mathbf{x}_c$ equals the sum of the links inside partition c.

The above can be written as the maximization of trace($\tilde{X}^T A \tilde{X}$), where the c-th column of $\tilde{X}$ equals $\tilde{\mathbf{x}}_c$; clearly $\tilde{X}^T \tilde{X} = I_k$. It is easy to verify that $\tilde{X}$ equals $\tilde{Y}$ from the previous section when all weights are set to one. It follows that if $K = A$ and $W = I$, the trace maximization for weighted kernel k-means (3) is equal to the trace maximization for ratio association.

Since the trace maximizations are mathematically equivalent, we can run weighted kernel *k*-means on the adjacency matrix to monotonically increase the ratio association in the graph. However, it is important to note that the adjacency matrix should be positive definite to ensure that it is a valid kernel matrix and can be factored as $\Phi^T\Phi$, thus allowing us to guarantee that the kernel k-means objective function decreases at each iteration. If the matrix is not positive definite, then there is no such guarantee (however, positive definiteness is a sufficient but not necessary condition for convergence). We will see how to satisfy this requirement in Section IV-D.

Note that this equivalence was discussed in [11]; no efficient algorithms beyond standard spectral techniques were developed in the paper to exploit the equivalence.

Ratio Cut. Next we consider the ratio cut problem:

$$\min \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{|\mathcal{V}_c|}.$$

Define a diagonal degree matrix D with $D_{ii} = \sum_{j=1}^{n} A_{ij}$. Using the indicator vector $\mathbf{x}_c$ from before, we can easily verify that the above objective function can be rewritten as:

$$\min \left\{ \sum_{c=1}^{k} \frac{\mathbf{x}_c^T (D - A) \mathbf{x}_c}{\mathbf{x}_c^T \mathbf{x}_c} = \sum_{c=1}^{k} \tilde{\mathbf{x}}_c^T (D - A) \tilde{\mathbf{x}}_c = \sum_{c=1}^{k} \tilde{\mathbf{x}}_c^T L \tilde{\mathbf{x}}_c \right\},$$

where $\tilde{\mathbf{x}}_c$ is defined as before, and the matrix $L \equiv D - A$ is known as the Laplacian of the graph. Hence, we may write the problem as a minimization of trace($\tilde{X}^T L \tilde{X}$). Consider the matrix $I - L$, and note that trace($\tilde{X}^T (I - L) \tilde{X}$) = trace($\tilde{X}^T \tilde{X}$) − trace($\tilde{X}^T L \tilde{X}$). Since $\tilde{X}$ is orthonormal, trace($\tilde{X}^T \tilde{X}$) = k, so maximizing trace($\tilde{X}^T (I - L) \tilde{X}$) is equivalent to the minimization of trace($\tilde{X}^T L \tilde{X}$).

Putting this all together, we have arrived at an equivalent trace maximization problem for ratio cut: minimizing the ratio cut is equivalent to maximizing trace($\tilde{X}^T (I - L) \tilde{X}$). Since $\tilde{X}$ equals $\tilde{Y}$ from the previous section when all weights are equal to 1, this corresponds exactly to unweighted kernel *k*-means, except that the matrix K is $I - L$. Note that while L is known to be positive definite $I - L$ can be indefinite. We will see how to deal with this issue in Section IV-D.

Kernighan-Lin Objective. The Kernighan-Lin (K-L) graph clustering objective follows easily from the ratio cut objective. For the case of K-L clustering, we maintain equally sized partitions, and hence the only difference between the ratio cut and K-L clustering is that each of the $\mathbf{x}_c$ indicator vectors is constrained to have exactly $|\mathcal{V}|/k$ non-zero entries. If we start with equally sized partitions, an incremental weighted kernel *k*-means algorithm (where we only swap points, or perform a chain of swaps, to improve the objective function) can be run to simulate the Kernighan-Lin algorithm. We discuss this in further detail in Section V-D.

Normalized Cut. We noted earlier that the normalized cut problem is equivalent to the normalized association problem; i.e., the problem can be expressed as:

$$\max\left\{\sum_{c=1}^{k}\frac{\text{links}(\mathcal{V}_c,\mathcal{V}_c)}{\text{degree}(\mathcal{V}_c)}=\sum_{c=1}^{k}\frac{\mathbf{x}_c^TA\mathbf{x}_c}{\mathbf{x}_c^TD\mathbf{x}_c}=\sum_{c=1}^{k}\tilde{\mathbf{x}}_c^TA\tilde{\mathbf{x}}_c\right\},$$

where $\tilde{\mathbf{x}}_c=\mathbf{x}_c/(\mathbf{x}_c^TD\mathbf{x}_c)^{1/2}$.

The above may be re-written as trace$(Y^TD^{-1/2}AD^{-1/2}Y)$, where $Y=D^{1/2}\tilde{X}$, and is orthonormal. In this case, we set the weighted kernel *k*-means weight matrix W in (3) to D and the matrix K to $D^{-1}AD^{-1}$. It can be verified that $Y=\tilde{Y}$ in this case; thus the objective functions for weighted kernel *k*-means and normalized cut are exactly equivalent. If the matrix K is positive definite, we have a way to iteratively minimize the normalized cut using weighted kernel *k*-means. Note that the seemingly simpler choice of $W=D^{-1}$ and $K=A$ does not result in a direct equivalence since Y and $\tilde{Y}$ are functions of the partition as well as the weights (and so $Y\neq\tilde{Y}$ if $W=D^{-1}$ and $K=A$).

The equivalence between normalized cuts and a weighted k-means formulation was first shown in [8] and our earlier paper [9].

General Weighted Graph Cuts/Association. More generally, the weighted association problem can be expressed as a trace maximization problem:

$$\max\left\{\sum_{c=1}^{k}\frac{\text{links}(\mathcal{V}_c,\mathcal{V}_c)}{w(\mathcal{V}_c)}=\sum_{c=1}^{k}\frac{\mathbf{x}_c^TA\mathbf{x}_c}{\mathbf{x}_c^TW\mathbf{x}_c}=\sum_{c=1}^{k}\tilde{\mathbf{x}}_c^TA\tilde{\mathbf{x}}_c\right\},$$

where $\tilde{\mathbf{x}}_c=\mathbf{x}_c(\mathbf{x}_c^TW\mathbf{x}_c)^{-1/2}$. With $Y=W^{1/2}\tilde{X}$, this simplifies to

$$WAssoc(G)=\max_Y \text{trace}(Y^TW^{-1/2}AW^{-1/2}Y). \qquad (4)$$

Our analysis easily extends to the $WCut$ problem. Using the same notation as before:

$$\begin{aligned}WCut(G) &= \min\sum_{c=1}^{k}\frac{\mathbf{x}_c^T(D-A)\mathbf{x}_c}{\mathbf{x}_c^TW\mathbf{x}_c}=\min\sum_{c=1}^{k}\tilde{\mathbf{x}}_c^TL\tilde{\mathbf{x}}_c\\ &= \min_Y \text{trace}(Y^TW^{-1/2}LW^{-1/2}Y).\end{aligned}$$

The $WCut$ problem can be expressed as $WAssoc$ by noting that

$$\begin{aligned}&\text{trace}(Y^TW^{-1/2}(W-L)W^{-1/2}Y)\\ =\;& k-\text{trace}(Y^TW^{-1/2}LW^{-1/2}Y).\end{aligned}$$

Hence, optimizing $WAssoc$ on the matrix $W-L$ is equivalent to optimizing $WCut$ on A.

It is easy to see that the matrix Y in this section is identical to $\tilde{Y}$ from Section 4.1. Setting the kernel matrix K to $W^{-1}AW^{-1}$, the trace maximization for weighted kernel *k*-means in (3) is seen to equal trace$(\tilde{Y}^TW^{-1/2}AW^{-1/2}\tilde{Y})$, which is *exactly* the trace maximization for weighted graph association from (4). In the other direction, given a kernel matrix K and a weight matrix W, define an adjacency matrix $A=WKW$ to obtain the equivalence. Thus, we see how to map from one problem to the other.

Computationally, the weighted kernel k-means algorithm can be expressed purely in graph-theoretic terms. Recall (2), the formula for computing the distance from a point to the centroid of cluster c:

$$K_{ii}-\frac{2\sum_{\mathbf{a}_j\in\pi_c}w_jK_{ij}}{\sum_{\mathbf{a}_j\in\pi_c}w_j}+\frac{\sum_{\mathbf{a}_j,\mathbf{a}_l\in\pi_c}w_jw_lK_{jl}}{(\sum_{\mathbf{a}_j\in\pi_c}w_j)^2}.$$

We can ignore the first term as a constant. If we consider running weighted kernel k-means for the weighted graph association objective, we use $K=W^{-1}AW^{-1}$ as our kernel matrix, and so we can rewrite this distance calculation as:

$$\frac{\sum_{\mathbf{a}_j,\mathbf{a}_l\in\pi_c}A_{jl}}{(\sum_{\mathbf{a}_j\in\pi_c}w_j)^2}-\frac{2\sum_{\mathbf{a}_j\in\pi_c}A_{ij}}{w_i\sum_{\mathbf{a}_j\in\pi_c}w_j}=\frac{\text{links}(\mathcal{V}_c,\mathcal{V}_c)}{w(\mathcal{V}_c)^2}-\frac{2\text{links}(v_i,\mathcal{V}_c)}{w_i\cdot w(\mathcal{V}_c)}.$$

The above expression can be used to compute step 2 in Algorithm 1, and so the resulting weighted kernel k-means algorithm is purely graph-theoretic. Note that simple extensions of the above formula can be used if shifting is performed to enforce positive definiteness (discussed below in Section IV-D).

C. The Spectral Connection

A standard result in linear algebra [18] states that if we relax the trace maximizations in (4) such that Y is an arbitrary orthonormal matrix, then the optimal Y is of the form V_kQ, where V_k consists of the leading k eigenvectors of $W^{-1/2}AW^{-1/2}$ and Q is an arbitrary $k\times k$ orthogonal matrix. As these eigenvectors are not indicator vectors, postprocessing (or rounding) of the eigenvectors is needed to obtain a discrete clustering. Nearly all spectral clustering objectives can be viewed as special cases of the general trace maximization problem in (4). The corresponding spectral algorithms compute and use the leading k eigenvectors of $W^{-1/2}AW^{-1/2}$ to optimize graph clustering objectives, such as ratio cut [5] and normalized cut [6], [7], [16]. Our equivalence shows that spectral solutions are not necessary, and the objectives may be directly optimized by kernel k-means.

Spectral methods typically perform well because they compute the globally optimal solution of a relaxation to the original clustering objective. However, these methods can be expensive on very large graphs. In contrast, kernel k-means is fast, but is prone to problems of poor local minima and sensitivity to initial starting partitions. In Section V, we will exploit the above analysis to develop a fast multilevel algorithm that overcomes such problems. This algorithm consistently outperforms spectral methods in terms of objective function value, as well as speed and memory usage. Before developing this algorithm, we briefly show how to enforce positive definiteness for graph clustering.

D. Enforcing Positive Definiteness

For weighted graph association, we define a matrix $K=W^{-1}AW^{-1}$ to map to weighted kernel *k*-means. However, when A is an arbitrary adjacency matrix, $W^{-1}AW^{-1}$ need not be positive definite and hence kernel k-means will not necessarily converge. In this section, we show how to avoid this problem by introducing an appropriate diagonal shift to K, and briefly discuss how such a diagonal shift affects the practical performance of weighted kernel *k*-means. This solution can be viewed as a generalization to the work in [11], where diagonal shifting was used in the unweighted case.

Given A, define $K'=\sigma W^{-1}+W^{-1}AW^{-1}$, where σ is a positive constant large enough to ensure that K' is positive definite. Since W^{-1} is a positive diagonal matrix, adding σW^{-1}

Objective	Node Weight	Kernel
Ratio Assoc.	1 for each node	$K = \sigma I + A$
Ratio Cut	1 for each node	$K = \sigma I - L$
Kernighan-Lin	1 for each node	$K = \sigma I - L$
Norm. Cut	Deg. of the node	$K = \sigma D^{-1} + D^{-1}AD^{-1}$

TABLE II
POPULAR GRAPH CLUSTERING OBJECTIVES AND CORRESPONDING WEIGHTS AND KERNELS FOR WEIGHTED KERNEL k-MEANS GIVEN ADJACENCY MATRIX A

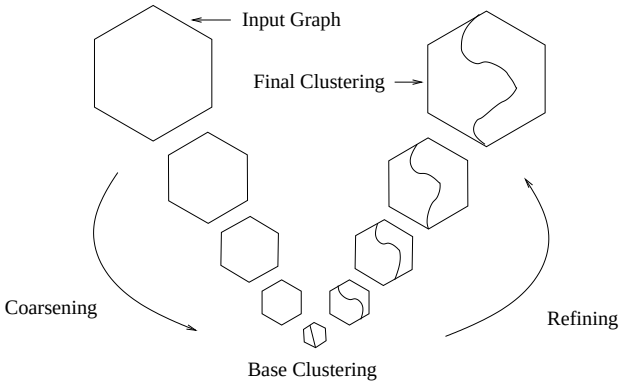

Fig. 1. Overview of the multilevel algorithm (for $k = 2$).

adds positive entries to the diagonal of $W^{-1}AW^{-1}$. Substituting K' for K in (3), we get:

$$\begin{aligned}\text{trace}(\tilde{Y}^T W^{1/2} K' W^{1/2} \tilde{Y}) &= \text{trace}(\tilde{Y}^T W^{1/2} \sigma W^{-1} W^{1/2} \tilde{Y}) \\ &\quad + \text{trace}(\tilde{Y}^T W^{-1/2} A W^{-1/2} \tilde{Y}) \\ &= \sigma k + \text{trace}(\tilde{Y}^T W^{-1/2} A W^{-1/2} \tilde{Y}).\end{aligned}$$

Hence, the maximizer $\tilde{Y}$ using K' is identical to that of the weighted association problem in (4), except that K' is constructed to be positive definite. Running weighted kernel *k*-means on K' results in monotonic optimization of the weighted association objective.

A similar approach can be used for the weighted cut problem. In Section IV-B, we showed that $WCut$ on the adjacency matrix A is equivalent to $WAssoc$ on $W - L$. Hence, if we let $A' = W - L$, it follows that defining $K' = \sigma W^{-1} + W^{-1}A'W^{-1}$ for large enough σ yields a positive definite kernel. In Table II, the weights and kernels for various graph objectives are summarized. Though adding a diagonal shift does not change the global optimizer, it is important to note that adding too large a shift may result in a decrease in quality of the clusters produced by Algorithm 1. This is due to points becoming closer to their current centroids and farther from other centroids as σ increases. See [19] for a detailed analysis.

V. THE MULTILEVEL ALGORITHM

Typically, the best algorithms for optimizing weighted graph clustering objectives use spectral clustering methods. In this section, we use the theoretical equivalence in Section IV-B to develop a new kernel-based multilevel clustering algorithm.

The framework of our multilevel algorithm is similar to that of Metis [12], a popular multilevel graph clustering algorithm for optimizing the Kernighan-Lin objective. Figure 1 provides a graphical overview of the multilevel framework. We assume that we are given an input graph $G_0 = (\mathcal{V}_0, \mathcal{E}_0, A_0)$, along with the number of desired partitions. Below we describe our multilevel algorithm in terms of its three phases: coarsening, base clustering, and refinement.

A. Coarsening Phase

Starting with the initial graph G_0, the coarsening phase repeatedly transforms the graph 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}, sets of nodes in G_i are combined to form supernodes in G_{i+1}. When combining a set of nodes into a single supernode, the edge weights out of the supernode are taken to be the sum of the edge weights out of the original nodes.

A popular approach for multilevel coarsening (and one used by Metis) is to use heavy edge coarsening. This approach works as follows: given a graph, start with all nodes unmarked. Visit each vertex in a random order. For each vertex x, if x is not marked, merge x with the unmarked vertex y that corresponds to the highest edge weight among all edges between x and unmarked vertices. Then mark x and y. If all neighbors of x have been marked, mark x and do not merge it with any vertex. Once all vertices are marked, the coarsening for this level is complete.

This coarsening procedure works well for the Kernighan-Lin objective, but we generalize it to a *max-cut coarsening* procedure to make it effective for a wider class of objectives. Given a vertex x, instead of merging using the criterion of heavy edges, we instead look for the unmarked vertex y that maximizes

$$\frac{e(x,y)}{w(x)} + \frac{e(x,y)}{w(y)}, \tag{5}$$

where $e(x, y)$ corresponds to the edge weight between vertices x and y and $w(x), w(y)$ are the weights of vertices x and y, respectively. These vertex weights correspond to the weights in Section IV. For example, in normalized cut, the weight of a vertex is its degree, and (5) reduces to the normalized cut between x and y. For ratio association, (5) simplifies to the heaviest edge criterion, since the weight is 1 for all vertices.

B. Base 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 $5k$ nodes, where k is the number of desired clusters. At this point, we perform a base (or initial) clustering by directly clustering the coarsest graph.

One base clustering approach is the region growing algorithm of Metis [12]. This algorithm clusters the base graph by selecting random vertices and growing regions around the vertices in a breadth-first fashion to form clusters. Since the quality of the resulting clusters depends on the choice of the initial vertices, the algorithm is run several times with different starting vertices and the best clustering is selected. This base clustering method is extremely efficient, though it tends to generate clusters of nearly equal size. Alternatively, we have found that an effective base clustering is provided by a state-of-the-art spectral clustering algorithm. This spectral algorithm generalizes the normalized cut algorithm of Yu and Shi [16] to work with arbitrary weights. As

a result, the base clustering may be optimized for different graph clustering objectives. Further details are available in our technical report [19]. Since the coarsest graph is significantly smaller than the input graph, spectral methods are adequate in terms of speed.

In addition to region growing and spectral clustering, we have also tested a third approach for base clustering—the bisection method. In this method, we bisect the coarsest graph into two clusters, then break up these two clusters into four clusters, and so on until k clusters are obtained. When bisecting a cluster, we run the multilevel algorithm with $k = 2$ to a coarsest level of 20 nodes, and use kernel k-means with random initialization at the lowest level. The region growing and bisection methods require no eigenvector computation, which makes them simpler and more appealing than spectral initialization. In Section VI, we compare different base clustering methods in terms of quality and speed.

C. Refinement

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 graph used in level $i-1$ of the coarsening phase). The clustering in G_i induces a clustering in G_{i-1} as follows: if a supernode in G_i is in cluster c, then all nodes in G_{i-1} formed from that supernode are in cluster c. This yields an initial clustering for the graph G_{i-1}, which is then improved using a refinement algorithm. Note that we also run the refinement algorithm on the coarsest graph. The multilevel algorithm terminates after refinement is performed on the original graph G_0. Since we have a good initial clustering at each level, the refinement usually converges very quickly, making this procedure extremely efficient.

Our multilevel algorithm uses weighted kernel *k*-means for the refinement step. Depending on the graph clustering objective to be optimized, we can appropriately set the weights and kernel at each step of the refinement given the adjacency matrix for the current level. At all levels except the coarsest level, the initialization for weighted kernel *k*-means is taken to be the clustering induced by the previous level.

We have optimized our implementation of weighted kernel k-means in several ways for maximum efficiency. For example, we have found that using only "boundary" points for the weighted kernel k-means algorithm speeds up computation considerably with little loss in cluster quality. When running kernel k-means, most points that are swapped from one cluster to another cluster in one iteration lie on the cluster boundary; that is, these nodes contain an edge to a node in another cluster. When determining which points to move from one cluster to another in weighted kernel k-means, we have the option of only considering such boundary points to speed up the computation.

Moreover, we can compute distances in the kernel space efficiently. We can precompute $w(\mathcal{V}_c)$ and links$(\mathcal{V}_c, \mathcal{V}_c)$ for every cluster to save computation time (see the end of Section IV-B), and we can compare distances without having to explicitly form the kernel matrix or use any floating point division (and thus our code only uses integer arithmetic if the initial edge weights are integers).

D. Local Search

A common problem when running standard batch kernel *k*-means is that the algorithm has a tendency to be trapped into qualitatively poor local minima. An effective technique to counter this issue is to do local search by incorporating an incremental strategy. A step of incremental kernel *k*-means attempts to move a single point from one cluster to another in order to improve the objective function. For a single move, we look for the move that causes the greatest decrease in the value of the objective function. For a chain of moves, we look for a sequence of such moves. The set of local minima using local search is a superset of the local minima using the batch algorithm. Often this enables the algorithm to reach much better local minima. It has been shown in [20] that incremental *k*-means can be implemented efficiently. Such an implementation can be easily extended to weighted kernel *k*-means. In practice, we alternate between doing standard batch updates and incremental updates. Further details can be found in [20].

As shown earlier, the Kernighan-Lin objective can also be viewed as a special case of the weighted kernel *k*-means objective (the objective is the same as ratio cut except for the cluster size restrictions), but running weighted kernel *k*-means provides no guarantee about the size of the partitions. We may still optimize the Kernighan-Lin objective by using a local search approach based on swapping points: if we perform only local search steps via swaps during the weighted kernel *k*-means algorithm, then cluster sizes remain the same. An approach to optimizing Kernighan-Lin would therefore be to run such an incremental kernel *k*-means algorithm using chains of swaps. This approach is very similar to the usual Kernighan-Lin algorithm [14].

VI. Experimental Results

In this section, we present a number of experiments to show the efficiency and effectiveness of our multilevel algorithm, which we call Graclus.[1] The problem of graph clustering arises in various real-life pattern recognition applications; we examine problems in gene network analysis, social network analysis and image segmentation. These applications do not generally have ground truth clusters, but yield examples of large clustering problems that frequently arise in practice. We show that Graclus discovers meaningful clusters in all these varied applications, and outperforms spectral methods in terms of objective function value and computation time. Later, we perform experiments on some standard benchmark data sets which have been used for testing graph clustering algorithms to objectively compare our methods to spectral methods. In all experiments, we use a state-of-the-art spectral clustering algorithm based on [16] for comparison purposes. This algorithm uses a sophisticated postprocessing (or rounding) method for computing a discrete clustering from the eigenvectors, and empirically, it performed the best among various spectral clustering algorithms that we tested in terms of optimizing the objective function value. Note that the algorithm in [16] was developed only for the normalized cut criterion; we have adapted it to work for general weighted graph clustering objectives.

Due to the different base clustering methods, the option of local search, and the option of using all points or boundary points, Graclus has several potential variants. Except for the benchmark graphs in Section VI-D, we use region growing initialization, which we found overall to have the best tradeoff between speed and quality; in Section VI-D, we compare different base clustering

[1] Software for our multilevel algorithm is available at http://www.cs.utexas.edu/users/dml/Software/graclus.html

# clusters	4	8	16	32	64	128
Graclus_All	0	.009	.12	.21	3.89	16.78
Graclus_Bdry	0	.009	.13	.52	4.44	17.65
Spectral	0	.037	.13	.92	5.37	25.46

TABLE III
NORMALIZED CUT VALUES RETURNED BY GRACLUS AND THE SPECTRAL METHOD

methods with the spectral algorithm and Metis on a number of benchmark graphs in terms of computation time as well as objective function values. All experiments were performed on a Linux machine with a 2.4 GHz Pentium IV processor and 1 GB main memory.

A. Gene network analysis

We first consider the Mycobacterium tuberculosis gene network from [21], where nodes are genes and an edge exists between two genes if and only if these two genes are inferred to be functionally linked by two or more of the following computational methods: the Rosetta Stone method [22], the Phylogenetic profile method [23], the Operon method [24] and the Conserved Gene Neighbor method [25]. This network contains 1381 nodes and 9766 functional edges. The weight of an edge is defined to be the number of methods that infer the two associated nodes to have functional linkage, thus the weights range in value from 2 to 4. A spy plot of the sparse functional linkage matrix is shown in the left panel of Figure 2. We apply two variants of Graclus and the spectral algorithm on this graph using the normalized cut objective to generate 4, 8, 16, 32, 64 and 128 clusters. Table III shows the normalized cut values produced by the three algorithms—Graclus with all the points, Graclus with only the boundary points, and the spectral algorithm. In all cases, Graclus outperforms the spectral method in terms of the normalized cut value.

To visually demonstrate that genes strongly connected in the functional linkage graph are grouped together by Graclus, we rearrange the rows and columns of the gene matrix so that rows (and columns) in the same cluster are adjacent to one another using the results from Graclus with boundary points for $k = 128$. The right panel of Figure 2 shows the matrix after rearrangement. As can be seen, after clustering, most of the nonzeros are concentrated around the diagonal, suggesting a tight clustering.

We now discuss a sampling of the produced clusters to demonstrate their biological significance. The histidine biosynthesis pathway is known to contain 10 genes: Rv1599 (hisD), Rv1600 (hisC), Rv1601 (hisB), Rv1602 (hisH), Rv1603 (hisA), Rv1605 (hisF), Rv1606 (hisI2), Rv2121c (hisG), Rv2122c (hisI), and Rv3772 (hisC2) [21]. The pathway is shown in Figure 3A, where arrows denote the direction of biochemical reactions and each gene denotes the enzyme specified by the corresponding gene that is necessary to carry out the reactions. Graclus accurately uncovers this pathway—all 10 of these genes are found to be in one cluster. Another cluster discovered by the multilevel method contains 7 out of 8 genes involved in the ATP synthase complex: Rv1304 (AtpB), Rv1306 (AtpF), Rv1307 (AtpH), Rv1308 (AtpA), Rv1309 (AtpG), Rv1310 (AtpD), and Rv1311 (AtpC)

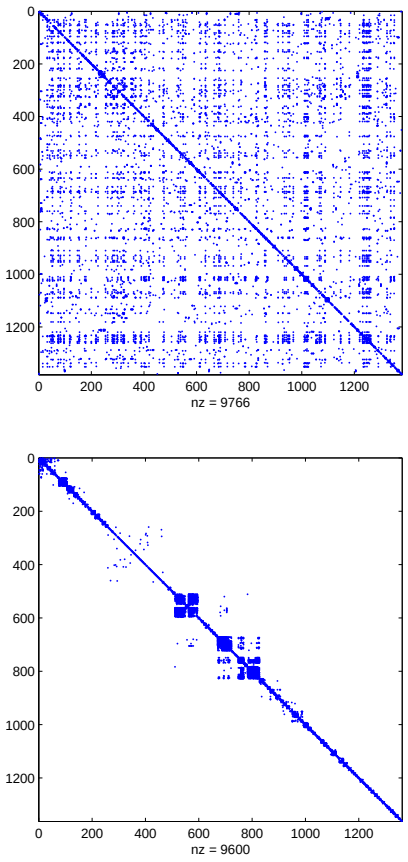

Fig. 2. Spy plots of the functional linkage matrix before and after clustering (128 clusters)—each dot indicates a non-zero entry.

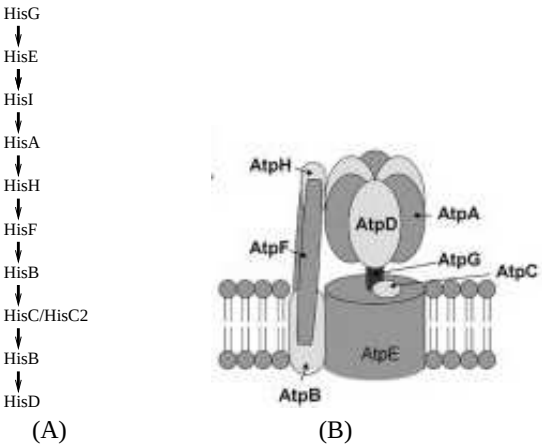

Fig. 3. (A) Histidine biosynthesis pathway, (B) ATP synthase multiprotein complex [21]

[21]. The eighth gene, RV1305 (AtpE), is not in the network since it is functionally linked to other genes only by the Operon method. Figure 3B illustrates the complex embedded in the membrane. As protons flow through the membrane, the cylinder-like AtpE and shaft-like AtpG rotate. The rotation of the shaft causes a conformational change in the subunits (AtpD and AtpA) of the orange-shaped head, where ATP is made.

We can also use the clustering results to infer previously uncharacterized gene functions. For example, one of the clusters discovered by our algorithm contains 16 genes: Rv0016c, Rv0017c, Rv0019c, Rv0050, Rv2152c, Rv2153c, Rv2154c, Rv2155c, Rv2156c, Rv2157c, Rv2158c, Rv2163c, Rv2165c, Rv2166c, Rv2981c and Rv3682. According to the Sanger M.tuberculosis web server[2], 12 of these genes have the annotation *murein sacculus and peptidoglycan*, a polymer that is composed of polysaccharide and peptide chains and is found in Mycobacterial cell walls. So we can infer that the remaining genes Rv0019c, Rv2154c, Rv2165c and Rv2166c have functions related to cell wall/envelope metabolism, which coincides with the results discovered in [21].

B. The IMDB Movie Data Set

The IMDB data set[3] contains information about movies, music bands, actors, movie festivals and events. By connecting actors with movies or events in which they participate, we form a large, sparse undirected graph with about 1.36 million nodes and 4.27 million edges.

Due to the size of the graph, we do not apply local search when running Graclus. We apply the spectral algorithm and Graclus to this data set and generate 2, 4, 8,..., 1024 clusters. We record the normalized cut and ratio association values along with computation times in Tables IV and V. Note that some cells in the tables are left empty because the spectral method cannot be run due to memory issues—storing k eigenvectors of this graph quickly becomes prohibitive. Comparing the values when the spectral method can be run, we see that in most cases Graclus produces better objective function values than the spectral algorithm: 4 out of 5 for ratio association and 4 out of 5 for normalized cut when running Graclus on the boundary points, and 5 out of 5 for both objectives when running on all the points. Interestingly, there is a significant improvement in ratio association values when using all points in Graclus as opposed to the boundary points.

Both variants of Graclus are much faster when k is large; for example, to produce 32 clusters using the normalized cut objective, Graclus with boundary points takes only about 40 seconds, while the spectral algorithm requires over 4400 seconds. Note that the spectral method for normalized cut takes many more iterations to converge than the spectral method for ratio association, yielding larger computation times.

In order to demonstrate the quality of clustering, we generate 5000 clusters so the clusters are sufficiently small. Note that it is impractical to run a spectral algorithm directly on this data set to produce 5000 clusters not only because computing 5000 eigenvectors of a graph of this size would be extremely slow but also because storing the eigenvectors would require about 24 GB of main memory. It takes approximately 12 minutes for Graclus using the normalized cut objective and boundary points to cluster this graph into 5000 clusters. The resulting cluster sizes range from 10 to 7999.

Documentaries
'Catch Me If You Can': Behind the Camera (2003)
The Making of Steven Spielberg's 'Jaws' (1995)
The Making of 'Jurassic Park' (1995)
The Making of 'Lost World' (1997)
AI: From Drawings to Sets (2002)
The Making of 'E.T. The Extra-Terrestrial' (1996)
The Making of 'The Color Purple' (2003)
Take Off: Making 'The Terminal' (2004)
The World of 'Minority Report': An Introduction (2002)
Making Close Encounters (1990)

TABLE VII
A SELECTION OF DOCUMENTARIES IN CLUSTER 1008

We briefly discuss a sampling of the produced clusters. Cluster 911 is of size 98 and contains "Harry Potter" movies and the actors in these movies. The left column of Table VI lists movies in the cluster, where we see five Harry Potter films. There are also three other documentary TV programs on Harry Potter in this cluster. The right column of Table VI 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 1008 is of size 197 and contains the filmmaker Steven Spielberg. Table VII list several documentaries in this cluster about films he has made. In addition, the cluster also contains a number of people who work with Spielberg, such as composer John Williams and cinematographer Janusz Kaminski, both of whom have worked on several of Spielberg's movies. Small clusters often contain one movie or movie series and cast members that acted only in this movie or movie series. For example, cluster 155 contains "Cruise Ship" (2002) and 9 of its cast members; cluster 2350 contains the short series "Festival número" (No. 1-12), shot in year 1965. Popular actors, directors or well-known movie festivals are associated with more people, so they often belong to larger clusters.

C. Image segmentation

Normalized cuts are often used for image segmentation, and spectral methods are typically used [16]. However, computing the eigenvectors of a large affinity matrix may be computationally expensive, especially if many eigenvectors are needed. In addition, Lanczos-type algorithms can sometimes fail to converge in a prespecified number of iterations, which occurred in some of our MATLAB runs on the image data set. However, Graclus can optimize the normalized cut using an eigenvector-free approach.

Due to lack of space, we cannot present detailed results on image segmentation, but provide an example to demonstrate that Graclus can successfully be used in this domain. We construct the affinity matrix by preprocessing the image in Figure 4 using code obtained from Stella Yu.[4] Then we apply Graclus and the spectral algorithm to the affinity matrix. Graclus with boundary points gives a normalized cut value of .0221, smaller than .0239,

[2]Annotations from http://www.sanger.ac.uk/Projects/M_tuberculosis/Gene_list/

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

[4]http://www.cs.berkeley.edu/~stellayu/code.html

Normalized cut values—lower cut values are better

# clusters	2	4	8	16	32	64	128	256	512	1024
Graclus_All	.040	.155	.410	1.01	3.52	9.92	25.90	63.38	125.60	261.76
Graclus_Bdry	.050	.186	.467	1.17	3.80	10.09	25.23	63.46	143.22	316.10
Spectral	.048	.213	.832	4.14	16.87	-	-	-	-	-

Computation time (in seconds)

Graclus_All	41.49	45.08	47.89	51.19	54.25	67.22	83.11	116.14	177.84	261.88
Graclus_Bdry	31.87	34.45	35.64	36.70	39.87	47.76	54.37	65.86	75.58	100.56
Spectral	163.92	301.39	518.60	1566.35	4410.10	-	-	-	-	-

TABLE IV

NORMALIZED CUT VALUES AND COMPUTATION TIME FOR A VARIED NUMBER OF CLUSTERS OF THE IMDB MOVIE DATA SET, USING TWO VARIANTS OF OUR MULTILEVEL ALGORITHM AND THE SPECTRAL METHOD

Ratio association values—larger association values are better

# clusters	2	4	8	16	32	64	128	256	512	1024
Graclus_All	45.16	105.71	184.47	327.97	623.20	1076.16	1822.21	2971.87	4712.56	7253.46
Graclus_Bdry	12.18	24.39	50.49	98.31	189.84	352.14	628.04	1064.83	1802.05	3197.86
Spectral	10.13	19.30	52.77	94.30	172.36	-	-	-	-	-

Computation time (in seconds)

Graclus_All	84.62	82.09	84.11	105.01	118.04	125.57	161.16	261.85	410.84	613.36
Graclus_Bdry	33.15	35.15	36.36	37.89	42.94	45.92	51.63	63.85	81.53	156.20
Spectral	25.07	27.83	74.41	158.74	448.84	-	-	-	-	-

TABLE V

RATIO ASSOCIATION VALUES AND COMPUTATION TIME FOR A VARIED NUMBER OF CLUSTERS OF THE IMDB MOVIE DATA SET, USING TWO VARIANTS OF OUR MULTILEVEL ALGORITHM AND THE SPECTRAL METHOD

Movies	Actors
Harry Potter and the Sorcerer's Stone (2001) Harry Potter and the Chamber of Secrets (2002) Harry Potter and the Prisoner of Azkaban (2004) Harry Potter and the Goblet of Fire (2005) Harry Potter and the Order of the Phoenix (2007) Harry Potter: Behind the Magic (2001 TV) Harry Potter und die Kammer des Schreckens: Das grosse RTL Special zum Film (2002 TV) J.K. Rowling: Harry Potter and Me (2002)	Daniel Radcliffe, Rupert Grint, Emma Watson, Peter Best, Sean Biggerstaff, Scott Fern, Joshua Herdman, Harry Melling, Matthew Lewis, Robert Pattinson, James Phelps, Oliver Phelps, Tom Felton, Devon Murray, Edward Randell, Jamie Waylett, Shefali Chowdhury, Katie Leung, Stanislav Ianevski, Jamie Yeates, Chris Rankin Bonnie Wright, Alfred Enoch

TABLE VI

A SELECTION OF MOVIES AND ACTORS IN CLUSTER 911

the normalized cut value obtained using the spectral algorithm. Figure 4 shows the segmentation of the sample image (into 3 segments) performed by Graclus.

D. Benchmark graph clustering

The previous subsections have looked at applications in pattern recognition. For an objective comparison, we now compare Graclus to the spectral method and the Metis software on standard benchmark graphs in terms of computation time as well as clustering quality. The following variants of Graclus are considered: GraclusK, GraclusB and GraclusS, which denote Graclus using region growing, bisection and spectral initialization respectively. For all variants, if local search is not used we append 0 to the name; otherwise we append 20 to indicate that a local search chain of length 20 is used. In all these experiments, boundary points were used for clustering.

We generate 128 clusters for all the graphs listed in Table VIII using Graclus and the spectral algorithm. Results using the bisection method are very similar to those using the region growing initialization, and thus are not presented here. Instead, we present results for GraclusK and GraclusS. Computation times are given in Table IX. Graclus clearly outperforms the spectral algorithm in terms of computation time. In many cases, Graclus is hundreds or even thousands of times faster than the spectral algorithm. For example, the spectral algorithm takes 636.82 seconds to cluster the "finance256" graph while GraclusK0 and GraclusK20 finish

Fig. 4. Segmentation of a sample image. The leftmost plot is the original image and each of the 3 plots to the right of it is a component (cluster) — body, tail and background. The normalized cut value for Graclus is .0221, smaller than .0239, the normalized cut value obtained using the spectral method.

Graph name	No. of nodes	No. of edges	Application
add32	4960	9462	32-bit adder
finance256	37376	130560	financial optimization
gupta2	62064	2093111	linear programming
memplus	17758	54196	memory circuit
pcrystk02	13965	477309	structural engineering
rajat10	30202	50101	circuit simulation
ramage02	16830	1424761	navier stokes and continuity equations

TABLE VIII
TEST GRAPHS FROM VARIOUS APPLICATIONS [26].

Computation time (in seconds) for normalized cut

	spectral	graclusS0	graclusS20	graclusK0	graclusK20
add32	37.86	2.25	2.43	**.02**	**.02**
finance256	636.82	2.33	2.34	**.23**	.32
gupta2	333.63	15.10	19.94	**14.18**	29.95
memplus	262.46	6.41	6.36	**.21**	.35
pcrystk02	45.30	4.25	3.48	**.33**	.50
rajat10	514.34	1.84	2.14	**.08**	.10
ramage02	45.05	3.46	3.46	**1.06**	1.50

Computation time (in seconds) for ratio association

add32	14.59	2.57	2.24	**.02**	.03
finance256	122.83	2.06	2.74	**.23**	.49
gupta2	323.43	**17.30**	**17.30**	18.67	27.33
memplus	28.35	6.99	6.34	**.17**	.79
pcrystk02	38.03	5.90	4.12	**.32**	.46
rajat10	479.55	2.52	2.45	**.08**	.13
ramage02	30.65	2.88	3.24	**.96**	1.29

TABLE IX
COMPUTATION TIME (IN SECONDS) USED BY THE SPECTRAL METHOD AND GRACLUS TO COMPUTE NORMALIZED CUT VALUES (TOP TABLE) AND RATIO ASSOCIATION VALUES (BOTTOM TABLE).

in .23 and .32 seconds, respectively. Among the Graclus variants, spectral initialization is slower than the initialization by bisection or region growing.

The quality results are shown in Figure 5. Since the normalized cut and ratio association values for the test graphs are quite different, we scale the values in Figure 5 using the corresponding values generated by the spectral algorithm. Graclus achieves better results than the spectral method in most cases (for example, all variants of Graclus give lower normalized cut values in 6 of the 7 graphs, and at least one variant of Graclus gives a higher ratio association value for each graph). The variant using spectral initialization generally gives results that are slightly better than the variant that uses region growing as the initialization (base clustering step). For example, in the case of the "memplus" graph, all variants of Graclus produce ratio association values at least 20% more than the spectral algorithm. Local search can sometimes significantly improve the performance of Graclus. For example, local search increases the ratio association value by 34% and decreases the normalized cut value by 10% with GraclusK on the "gupta2" graph .

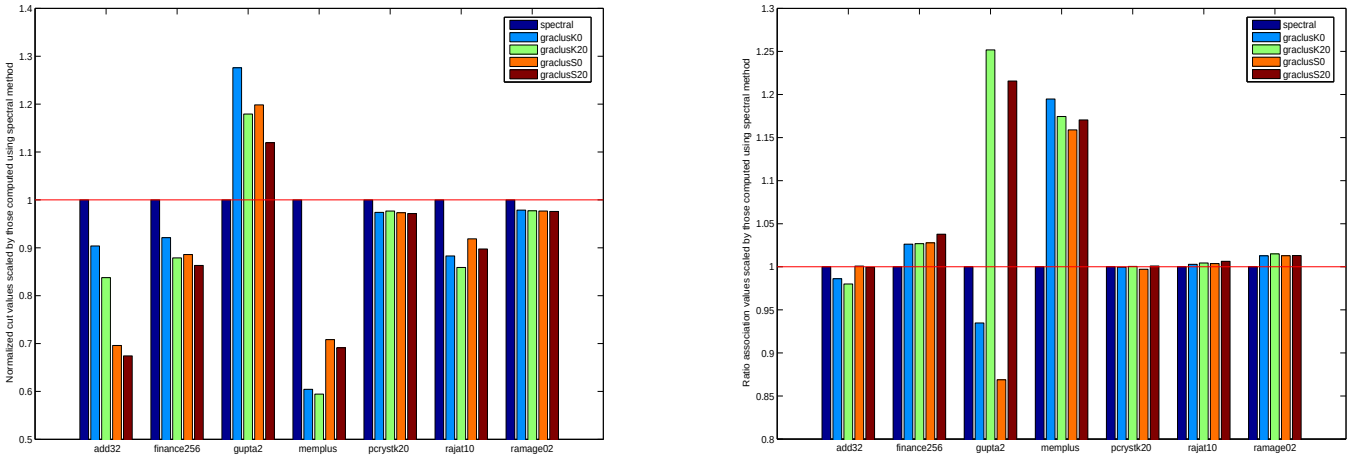

Fig. 5. Quality comparison between the spectral method and Graclus in terms of normalized cut (left panel) and ratio association (right panel) values for 128 clusters. The plot shows the effect of local search and different base clustering methods. All the values in each 5-bar group of the plot are scaled by the corresponding value generated using the spectral method. Note that in the *left* panel, bars *below* the horizontal line indicate that Graclus performs better; in the *right* panel, bars *above* the horizontal line indicate that Graclus performs better.

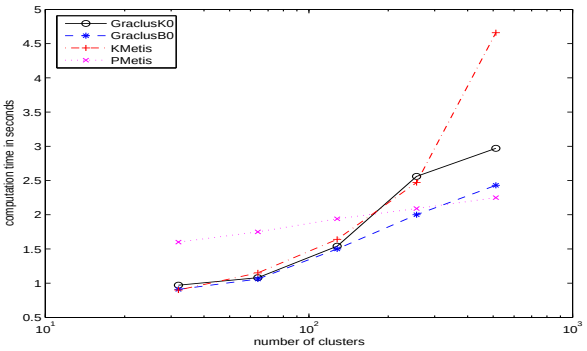

Fig. 6. Computation time comparison between Graclus and Metis on "ramage02"

We now compare Graclus with the Metis software in terms of scalability on a representative graph. Note that all our experiments show that Graclus consistently produces better normalized cut and ratio association values than Metis, which is not surprising since Metis minimizes the *edge cut* value instead of normalized cut or ratio association. Figure 6 shows the running times of Graclus and Metis for 32, 64, 128, 256 and 512 clusters of the "ramage02" graph. From the figure, we see that our Graclus algorithm is comparable to KMetis but runs faster than PMetis when the number of clusters is small; when the number of clusters becomes large, Graclus is comparable to PMetis but faster than KMetis.

In summary, all variants of Graclus are significantly faster than the spectral methods, and are comparable or better in terms of quality — among the variants, GraclusS20 generally gives the best clustering results. However, GraclusS20 is slightly slower than other variants of Graclus, such as GraclusK0, which is up to 2000 times faster than the spectral method. Furthermore, Graclus is comparable to Metis in terms of speed, and produces better weighted graph cuts.

VII. Related Work

There has been extensive work on various forms of spectral and graph clustering. The Kernighan-Lin heuristic has been used in graph clustering for decades [14], and many efficient algorithms have been proposed for locally optimizing the K-L objective. Some of the earliest work on spectral methods was done by Hall and Donath & Hoffman [3], [4]. The k-way ratio cut objective was used for circuit partitioning in [5], while normalized cuts were introduced for image segmentation in [6], and later extended to k-way partitioning in [16]. Kernel-based clustering using spectral methods was discussed in [27], [28]; the spectral algorithms discussed in [28] correspond to optimizing ratio association and ratio cut. Kernel k-means was introduced in [1] as an extension to standard Euclidean k-means. Other uses of kernels, most importantly in nonlinear component analysis, were also explored in [1]. Kernel-based learning methods have appeared in a number of other areas in the last few years, especially in the context of support vector machines [15].

The spectral relaxation of standard Euclidean k-means objective function was discussed in [10]. The formulation was based on unweighted k-means and no connection was made to graph cuts. A recent paper [11] also discusses a connection between k-means and spectral clustering in the unweighted case; however, normalized cuts were not amenable to the analysis in [11]. The latter paper also discussed the issue of enforcing positive definiteness for clustering. In [8], Bach and Jordan studied a cost function for spectral clustering that can be minimized to either learn the similarity matrix or the clustering. In a footnote, [8] mentioned a connection between k-means and normalized cut involving an $O(n^3)$ decomposition of a positive semi-definite similarity matrix. None of these papers consider the use of kernels

or generalized graph cuts. Weights, which play an important role in normalized cuts, are not discussed in [10] or [11]. More recently, [17] has considered connections between normalized cuts and k-means, as well as connections to doubly stochastic matrices and clustering with side information.

Our multilevel algorithm is based on Metis [12], a fast and popular graph partitioning algorithm. Metis is often cited as one of the best examples of a multilevel graph clustering algorithm. However, Metis and Chaco [13] are both restricted to the Kernighan-Lin heuristic and force nearly equally sized clusters. Some recent work has considered hierarchical spectral methods [29] to improve efficiency, though this work is limited to stochastic matrices (and thus applies only to the normalized cut objective); as future work, it may be possible to compare with, extend, or incorporate the methods discussed in this domain.

Preliminary versions of our work have appeared in [9], [30]. The current paper substantially extends the theoretical results of our earlier work, including generalizing graph cuts and methods for enforcing positive definiteness. The multilevel algorithm has been considerably improved over the version presented in [30], often by an order of magnitude in speed.

VIII. Conclusion

Spectral clustering has received considerable attention in the last few years as a powerful clustering method with varied applications. However, previous algorithms for spectral clustering objectives have relied on eigenvector computation, which can be prohibitive for very large data sets. In this paper, we have discussed a mathematical equivalence between a general spectral clustering objective and the weighted kernel k-means objective. Using this equivalence, we have designed a fast multilevel algorithm that outperforms spectral methods in terms of quality, speed and memory usage. Since special cases of this general spectral clustering objective include normalized cut, ratio cut and ratio association, our multilevel algorithm provides an eigenvector-free method for minimizing several popular objectives. The refinement step in our algorithm uses weighted kernel k-means, an algorithm which has heretofore received very little practical attention in the research community. Experimental results on a gene network, a social network and several benchmark graphs demonstrate the effectiveness of our multilevel algorithm.

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

References

[1] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[2] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Math., Stat. and Prob.*, 1967, pp. 281–296.

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

[4] K. M. Hall, "An r-dimensional quadratic placement algorithm," *Management Science*, vol. 11, no. 3, pp. 219–229, 1970.

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

[6] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, August 2000.

[7] A. Y. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. of NIPS-14*, 2001.

[8] F. Bach and M. Jordan, "Learning spectral clustering," in *Proc. of NIPS-17*. MIT Press, 2004.

[9] I. Dhillon, Y. Guan, and B. Kulis, "Kernel *k*-means, spectral clustering and normalized cuts," in *Proc. 10th ACM KDD Conference*, 2004, pp. 551–556.

[10] H. Zha, C. Ding, M. Gu, X. He, and H. Simon, "Spectral relaxation for *k*-means clustering," in *Neural Info. Processing Systems*, 2001.

[11] V. Roth, J. Laub, M. Kawanabe, and J. Buhmann, "Optimal cluster preserving embedding of non-metric proximity data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, 2003.

[12] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, 1999.

[13] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," Sandia National Laboratories, Tech. Rep. SAND93-1301, 1993.

[14] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[15] N. Cristianini and J. Shawe-Taylor, *Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge University Press, 2000.

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

[17] R. Zass and A. Shashua, "A unifying approach to hard and probabilistic clustering," in *10th IEEE Conf. on Computer Vision*, 2005.

[18] G. Golub and C. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 1989.

[19] I. Dhillon, Y. Guan, and B. Kulis, "A unified view of kernel *k*-means, spectral clustering and graph cuts," University of Texas at Austin, Tech. Rep. TR-04-25, 2004.

[20] 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*, 2002, pp. 131–138.

[21] M. Strong, T. Graeber, M. Beeby, M. Pellegrini, M. Thompson, T. Yeates, and D. Eisenberg, "Visualization and interpretation of protein networks in mycobacterium tuberculosis based on hierarchical clustering of genome-wide functional linkage maps," *Nucleic Acids Res.*, vol. 31(24), pp. 7099–7109, 2003.

[22] E. M. Marcotte, M. Pellegrini, H.-L. Ng, D. W. Rice, T. O. Yeates, and D. Eisenberg, "Detecting protein function and protein-protein interactions from genome sequences," *Science*, vol. 285, pp. 751–753, 1999.

[23] M. Pellegrini, E. M. Marcotte, M. J. Thompson, D. Eisenberg, and T. O. Yeates, "Detecting the components of protein complexes and pathways by comparative genome analysis: Protein phylogenetic profiles," *Proc. Natl. Acad. Sci.*, vol. 96, pp. 4285–4288, 1999.

[24] M. Strong, P. Mallick, M. Pellegrini, M. Thompson, and D. Eisenberg, "Inference of protein function and protein linkages in mycobacterium tuberculosis based on prokaryotic genome organization: a combined computational approach," *Genome Biol.*, vol. 4, pp. R59.1–16, 2003.

[25] R. Overbeek, M. Fonstein, M. D'Souza, G. Pusch, and N. Maltsev, "The use of gene clusters to infer functional coupling," *Proc Natl Acad Sci USA*, vol. 96(6), pp. 2896–2901, 1999.

[26] T. Davis, "University of Florida sparse matrix collection," vol. 92, no. 42, 1994; vol. 96, no. 28, 1996; vol. 97, no. 23, 1997. [Online]. Available: http://www.cise.ufl.edu/research/sparse/matrices

[27] M. Girolami, "Mercer kernel based clustering in feature space," *IEEE Transactions on Neural Networks*, vol. 13, no. 4, pp. 669–688, 2002.

[28] N. Cristianini, J. Shawe-Taylor, and J. Kandola, "Spectral kernel methods for clustering," in *Proc. of NIPS-14*, 2001.

[29] C. Chennubhotla and A. Jepson, "Hierarchical eigensolver for transition matrices in spectral methods," in *Proc. of NIPS 17*, 2004.

[30] I. Dhillon, Y. Guan, and B. Kulis, "A fast kernel-based multilevel algorithm for graph clustering," in *Proc. 11th ACM KDD Conference*, 2005, pp. 629–634.

Inderjit Dhillon is an Associate Professor of Computer Sciences at The University of Texas at Austin. His main research interests are in numerical analysis, data mining and machine learning. He is best known for his work on computational algorithms in these areas, in particular on eigenvalue computations, clustering, co-clustering and matrix approximations. Software based on his research on eigenvalue computations is now part of all state-of-the-art numerical software libraries. Inderjit received an NSF Career Award in 2001, a best paper award at the SIAM data mining conference in 2003, a University Research Excellence Award in 2005, and the SIAG/LA Prize in 2006. He received his B.Tech. degree from the Indian Institute of Technology at Bombay, and Ph.D. from the University of California at Berkeley. He is a member of the Association for Computing Machinery, the Institute of Electrical and Electronics Engineers, and the Society for Industrial and Applied Mathematics.

Yuqiang Guan received his B.S degree in computer science from the University of Science and Technology of China, in 1997. From 1997 to 1999, he studied as a graduate student in Western Michigan University, where he was the recipient of the Excellence in Research Award and Outstanding Graduate Research Award. He transfered to the University of Texas at Austin in 2000 and received his Ph.D. degree in computer science in May 2006. He is currently working in the core search team at Microsoft, Inc., Redmond, WA. Dr. Guan's research interests include large-scale text mining, clustering, information extraction and question answering.

Brian Kulis is a PhD student in computer science at the University of Texas at Austin. He graduated from Cornell University in 2003 with a BA in computer science and mathematics. His research interests are in machine learning and data mining, with specific interests in large-scale optimization, kernel methods, spectral methods, and convex analysis.