A Spectral Clustering Approach To Finding Communities in Graphs*

Scott White† and Padhraic Smyth†

Abstract

Clustering nodes in a graph is a useful general technique in data mining of large network data sets. In this context, Newman and Girvan [9] recently proposed an objective function for graph clustering called the Q function which allows automatic selection of the number of clusters. Empirically, higher values of the Q function have been shown to correlate well with good graph clusterings. In this paper we show how optimizing the Q function can be reformulated as a spectral relaxation problem and propose two new spectral clustering algorithms that seek to maximize Q. Experimental results indicate that the new algorithms are efficient and effective at finding both good clusterings and the appropriate number of clusters across a variety of real-world graph data sets. In addition, the spectral algorithms are much faster for large sparse graphs, scaling roughly linearly with the number of nodes n in the graph, compared to $O(n^2)$ for previous clustering algorithms using the Q function.

1 Introduction

Large complex graphs representing relationships among sets of entities are an increasingly common focus of scientific inquiry. Examples include social networks, Web graphs, telecommunication networks, semantic networks, and biological networks. One of the key questions in understanding such data is "How many communities are there and what are the community memberships"?

Algorithms for finding such communities, or automatically grouping nodes in a graph into clusters, have been developed in a variety of different areas, including VLSI design, parallel computing, computer vision, social networks, and more recently in machine learning. Good algorithms for graph clustering hinge on the quality of the objective function being used. A variety of different objective functions and clustering algorithms have been proposed for this problem, ranging from hierarchical clustering to max-flow/min-cut methods to methods based on truncating the eigenspace of a suitably-defined matrix. In recent years, much attention has been paid to spectral clustering algorithms (e.g., [11],[12],[14]) that, explicitly or implicitly, attempt to globally optimize cost functions such as the Normalized Cut measure [12]. The majority of these approaches attempt to balance the size of the clusters while minimizing the interaction between dissimilar nodes. However, for the types of complex heterogeneous networks that arise naturally in many domains, the bias that these approaches have towards clusters of equal size can be seen as a drawback. Furthermore, many of these measures, such as Normalized Cut, can not be used directly for selecting the number of clusters, k, since they increase (or decrease) monotonically as k is varied.

Recently, a new approach was developed by Newman and Girvan [9] to overcome limitations of previous measures for measuring community structure. They proposed the "modularity function" Q, which directly measures the quality of a particular clustering of nodes in a graph. It can also be used to automatically select the optimal number of clusters k, by finding the value of k for which Q is maximized, in contrast to most other objective functions used for graph clustering.

Let $G(V, E, W)$ be an undirected graph consisting of the set of nodes V, the set of edges E, and a symmetric weight matrix $W \in \Re^{n \times n}$, where n is the number of vertices. The weights $w_{ij} = w_{ji} = [W]_{ij}$ are positive if there is an edge between vertices v_i and v_j, and 0 otherwise. The modularity function Q can be defined as

$$Q(\mathcal{P}_k) = \sum_{c=1}^{k} \left[\frac{\mathcal{A}(V_c, V_c)}{\mathcal{A}(V, V)} - \left(\frac{\mathcal{A}(V_c, V)}{\mathcal{A}(V, V)} \right)^2 \right] \tag{1.1}$$

where $\mathcal{P}_k$ is a partition of the vertices into k groups and where $\mathcal{A}(V', V'') = \sum_{i \in V', j \in V''} w(i, j)$. Thus, $\mathcal{A}(V_c, V_c)$ measures the within-cluster sum of weights, $\mathcal{A}(V_c, V)$ measures the sum of weights over all edges attached to nodes in cluster c, and $\mathcal{A}(V, V)$ measures the sum of all edge weights in the graph. Considering binary weights for simplicity, the term $\frac{\mathcal{A}(V_c, V_c)}{\mathcal{A}(V, V)}$ is the empirical probability $\hat{p}_{c,c}$ that both ends of a randomly selected edge from G lie in cluster c. Similarly, $\frac{\mathcal{A}(V_c, V)}{\mathcal{A}(V, V)}$ is the empirical probability $\hat{p}_c$ that a specific end of an edge (either one), for a randomly selected edge, lies in cluster c. Thus, under an independence model, Q can be interpreted as a measure of the deviation between (a) the observed edge-cluster probabilities $\hat{p}_{c,c}$ and (b) what one would predict under an independence model: $\hat{p}_c^2$.

*The research in this paper was supported by the National Science Foundation under Grant IRI-9703120 as part of the Knowledge Discovery and Dissemination program. SW was also supported by a National Defense Science and Engineering Graduate Fellowship.

†Department of Computer Science, University of California, Irvine

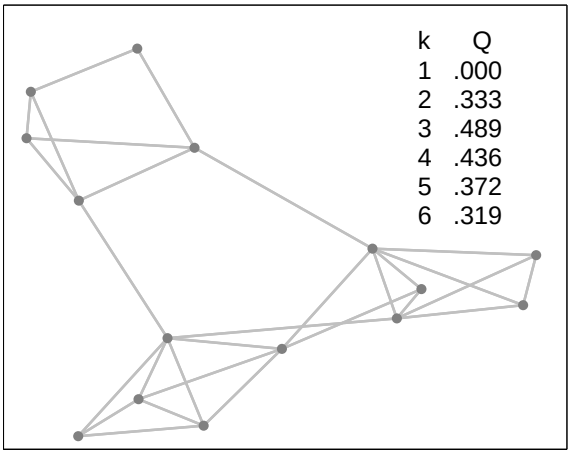

Figure 1: A toy graph showing Q values for different numbers of clusters.

Newman [10] and Newman and Girvan [9] showed across a wide variety of simulated and real-world graphs that larger Q values are correlated with better graph clusterings. In addition, they found that real-world unweighted networks with high community structure generally have Q values within a range from 0.3 to 0.7. Figure 1 shows an example of a simple toy graph with binary weights, where the structure of the graph visually suggests 3 clusters. Also shown are the maximum values of the Q function for different numbers of clusters k, and indeed Q is maximized for $k = 3$.

As pointed out in [10], if no edges exist that connect vertices across clusters then $Q = 1$, and conversely if the number of inter-cluster edges is no better than random then $Q = 0$. We have found empirically that the Q measure works well in practice in terms of both (a) finding good clusterings of nodes in graphs where community structure is evident, and (b) indicating what the appropriate number of clusters k is for such a graph.

In this paper we show how Newman's Q measure can be related to the broader family of spectral clustering methods. Specifically:

- We show how the problem of maximizing the modularity measure Q can be reformulated as an eigenvector problem involving a matrix we call the "Q-Laplacian." In this manner we link work on graph clustering using the Q measure to relevant work on spectral clustering (e.g., [11], [12],[14]).
- We use the eigenvector formulation of maximizing Q to derive two new spectral graph clustering algorithms. One of these algorithms directly seeks a global optimum of the Q function. The other algorithm is similar to Newman's agglomerative clustering algorithm [10], in that it attempts to maximize Q via local iterative improvement.
- We compare the new algorithms with Newman's algorithm on different graph data sets and empirically illustrate that:
  - the spectral approach to maximizing Q produces results that, in terms of cluster quality, are comparable or better than results from Newman's hierarchical algorithm, and
  - the proposed algorithms are linear per iteration in the number of nodes and edges in the graph, compared to quadratic complexity in the number of nodes for the original algorithm proposed by Newman [10].

2 Spectral Approaches to Maximizing the Q Function

Consider for the moment that the number of clusters, k, is fixed. We use the following strategy to address the problem of finding a partitioning that maximizes $Q(P_k)$ as follows:

1. Reformulate the problem of maximizing Newman's Q function as a discrete quadratic assignment problem.
2. Approximate the resulting assignment problem by relaxing it to a continuous one which can be solved analytically using eigen-decomposition techniques.
3. Compute the top $k-1$ eigenvectors of this solution to form a $k-1$-dimensional embedding of the graph into a Euclidean space. Use "hard-assignment" geometric clustering (the k-means algorithm) on this embedding to generate a clustering P_k.

Below we outline each of these steps. In the section which follows, Section 3, we then describe computational details for two proposed clustering algorithms based on this approach.

2.1 Quadratic Assignment We assume G is simple, i.e., G contains no self-loops nor parallel edges, and is connected, i.e., there is a path from any vertex to any other vertex. Let $D \in \Re^{n\times n}$ be a diagonal matrix having d_i in the ith diagonal entry and 0 everywhere else, where $d_i = \sum_j w_{ij}$. We denote the diagonal matrix derived from an $n \times n$ matrix X as $\text{diag}(X) \in \Re^{n\times n}$ where $[\text{diag}(X)]_{ij} = [X]_{ij}$ if $i = j$ and 0 otherwise. Finally, let $\text{tr}(X) = \sum_i [X]_{ii}$ be the trace of matrix X.

To simplify notation, we rewrite Q as follows:

$$Q(P_k) \propto \sum_{c=1}^{k} \left[\mathcal{A}(V,V)\mathcal{A}(V_c,V_c) - \mathcal{A}(V_c,V)^2\right] \tag{2.2}$$

Given a k-partition $\mathcal{P}_k$, define a corresponding $n \times k$ assignment matrix $X = [\mathbf{x_1}, \ldots, \mathbf{x_k}]$ with $x_{ic} = 1$ if $v_i \in V_c$, and $x_{ic} = 0$ otherwise, for $1 \le c \le k$. Observe that since each vertex can only be in one cluster, $X\mathbf{1_k} = \mathbf{1_n}$. We can reformulate Q in terms of the assignment matrix X as follows:

$$\begin{aligned} Q(P_k) &\propto \sum_{c=1}^{k}\left[\text{vol}_G \sum_{i,j=1}^{n} w_{ij}x_{ic}x_{jc} - \left(\sum_{i,j=1}^{n} w_{ij}x_{ic}\right)^2\right] \\ &= \sum_{c=1}^{k}\left[\text{vol}_G \mathbf{x_c}^T W \mathbf{x_c} - \left(\sum_{i=1}^{n} d_i x_{ic}\right)^2\right] \\ &= \sum_{c=1}^{k}\left[\text{vol}_G \mathbf{x_c}^T W \mathbf{x_c} - \left(\mathbf{d^T x_c}\right)^2\right] \end{aligned}$$

where $\mathbf{d} \in \Re^{n\times 1}$ such that component d_i equals the weighted degree of vertex i and we rewrite $\mathcal{A}(V,V)$ as vol_G, the volume of graph G. Thus,

$$\begin{aligned} Q(P_k) &\propto \sum_{c=1}^{k}\left[\text{vol}_G \mathbf{x_c^T} W \mathbf{x_c} - \mathbf{x_c^T d d^T x_c}\right] \\ &= \sum_{c=1}^{k}\left[\text{vol}_G \mathbf{x_c^T} W \mathbf{x_c} - \mathbf{x_c^T} \mathcal{D} \mathbf{x_c}\right] \end{aligned}$$

where $\mathcal{D} = \mathbf{dd^T}$. Since for any matrix A and assignment matrix X, $\text{tr}(X^T AX) = \sum_{c=1}^{k}\left[\mathbf{x_c^T} A \mathbf{x_c}\right]$, we can further reduce Q as follows:

$$\begin{aligned} Q(P_k) &\propto \text{vol}_G \text{tr}\left(X^T W X\right) - \text{tr}\left(X^T \mathcal{D} X\right) \\ &= \text{tr}\left(X^T(\text{vol}_G W - \mathcal{D})X\right) \\ &= \text{tr}\left(X^T(\mathcal{W} - \mathcal{D})X\right) \end{aligned}$$

where $\mathcal{W} = \text{vol}_G W$. The problem of maximizing Q can then be expressed as:

$$\max_X \left\{\text{tr}\left(X^T(\mathcal{W} - \mathcal{D})X\right)\right\} \text{ s.t. } X^T X = M \tag{2.3}$$

where $M \in \Re^{k\times k}$ is a diagonal matrix with diagonal entry $[M]_{ii} = |V_i|$, where $|V_i|$ is the number of nodes in cluster V_i.

2.2 Spectral Relaxation Finding an assignment matrix X which maximizes (2.3) is NP-complete. To address this we can attempt to derive a good approximation by relaxing the discreteness constraints that the $X_{ij} \in \{0, 1\}$, so that instead the $X_{ij} \in \Re^1$. This transforms the discrete optimization problem into one that is continuous. To find the optimal relaxed X, take the derivative of the following expression with respect to X:

$$\text{tr}\left(X^T(\mathcal{W} - \mathcal{D})X\right) + (X^T X - M)\Lambda \tag{2.4}$$

where $\Lambda \in \Re^{k\times k}$ is the diagonal matrix of Lagrangian multipliers. Setting this equal to 0 and rearranging terms we have:

$$L_Q X = X\Lambda \tag{2.5}$$

where $L_Q = \mathcal{D} - \mathcal{W}$ and we refer to this diagonal matrix as the "Q-Laplacian". Aside from noting its similarity in form to the standard Laplacian, we observe that this is a standard matrix eigenvalue problem which can be solved using standard eigendecomposition methods. Furthermore, had we normalized the original matrix W so that all rows sum to one and had we also added back in the normalization constant that we took out from equation (2.2) then we would have the following eigenvalue equation:

$$\mathcal{L}_Q X = X\Lambda \tag{2.6}$$

where $\mathcal{L}_Q = \frac{1}{n^2}E - \frac{1}{n}W'$ where E is a matrix of all ones and W' is the matrix W normalized so all rows sum to one. The first term of this equation can be seen as a damping term that ensures that there are edges between all of the nodes of very small weight and the second term is the original weight matrix after scaling and normalization. As $n \to \infty$, the first term will approach 0 much faster than the second term, and hence will play a negligible role in determining the eigenspace of the matrix.

Thus, for even moderately large values of n, it seems reasonable that W' will provide a close approximation to $\mathcal{L}_Q$, which we refer to as the "normalized Q Laplacian."[1] In this paper, we will adopt the simplest method

[1]The minus sign and the constant $\frac{1}{n}$ do not impact the resulting eigenspace.

for normalizing a matrix so its rows sum to one, namely, to left multiply the matrix W by D^{-1}. The advantage of using $W' = D^{-1}W$ as an approximation to $\mathcal{L}_Q$ is that it is easy to compute, it is well studied, especially in relation to Markov chains where it is known as the transition matrix, and it retains sparsity so we can use fast methods for eigendecomposing a sparse matrix.

The final step in this framework is to iterate over different values of k, to search for the best clusterings (highest $Q(P_k)$ scores). For each k, we try to find the optimal partitioning, i.e., a "hard-assignment" of the nodes to k clusters, based on clustering the rows of the matrix X.

3 Two New Graph Clustering Algorithms

In this section, we propose two new algorithms for clustering graphs that build on insights developed in the previous section.

3.1 Computing the embedding Assume that we are seeking up to a maximum of K clusters and that we have a weight matrix $W \in \Re^{nxn}$. Both of our proposed algorithms below begin by computing the top $K-1$ eigenvectors (ignoring the trivial all-ones eigenvector) corresponding to Equation 2.6. Specifically:

1. Compute the transition matrix $\mathcal{M} = D^{-1}W$
2. Compute the eigenvector matrix $U_K = [\mathbf{u_1}\mathbf{u_2}\ldots\mathbf{u_{K-1}}]$ from $\mathcal{M}$ using a sparse eigenvector decomposition method such as a variant of the Lanczos method or subspace iteration.

In the experimental results in this paper we compute the $K-1$ eigenvectors using the Implicitly Restarted Lanczos Method (IRLM) [2]. If one makes the conservative assumption that there are $O(K)$ extra Lanczos steps, then the IRLM has worse-case time complexity of $O(mKh + nK^2h + K^3h)$ where m is the number of edges in the graph, and h is the number of iterations required until convergence. For sparse graphs, where $m \sim n$, and where $K \ll n$, we found the IRLM to be extremely fast, taking near linear time with respect to the number of nodes n.

In the algorithms below, we initialized k-means so that the starting centroids were chosen to be as close to orthogonal as possible. Initializing k-means in this way does not change the time-complexity but can significantly help to improve the quality of the clusterings, as discussed in [11], while at the same reducing the need for multiple random restarts. In addition, both algorithms below can be run for any range of k values between a lower bound $k_{\min}$ and an upper bound $k_{\max}$. When not stated otherwise, we will assume in what follows that we have $k_{min} = 1$ and $k_{max} = K$. In the case where $k = 1, Q = 0$ and the cluster is just all the vertices in the graph.

3.2 Algorithm Spectral-1 This algorithm takes as input an eigenvector matrix U_K, and consists of the following steps:

1. For each value of k, $2 \le k \le K$:
   (a) Form the matrix U_k from the first $k-1$ columns of U_K.
   (b) Scale the rows of U_k using the l^2-norm so they all have unit length
   (c) Cluster the row vectors of U_k using k-means or any other fast vector-based clustering algorithm. For $k = 1$, the cluster is just the graph itself.
2. Pick the k and the corresponding partition that maximizes $Q(P_k)$.

This algorithm is similar in spirit to the one developed in [11]. Both algorithms embed the input graph into a Euclidean space by eigendecomposing a suitable matrix and then cluster the embedding using a geometric clustering algorithm. We experimentally validated the claim made in [11] that row-normalizing the matrix of eigenvectors, so that the row vectors are projected onto the unit hypersphere, gives much higher quality results. The Spectral-1 algorithm is different in three key respects to this earlier work (in addition to the minor ontological point that our framework is designed to cluster graphs while theirs is designed to cluster real-valued points):

1. Whereas in [11] the matrix that is eigendecomposed, $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, was implicitly chosen to optimize the Normalized Cut, our algorithm is explicitly designed to optimize the modularity Q.
2. Our algorithm has a natural method for model-selection, the Q measure, which is the same objective function our embedding is based on. Since Normalized Cut is biased by the size of k, it can not be used for choosing the best k.
3. Our algorithm does not require an extra step of model selection to ensure: a) the edge weights are scaled correctly and b) the graph is sparsified. If links are not sparsified in the algorithm in [11], the time complexity is $O(n^3)$.

Table 1: Sample clusters found for the WordNet data.

Hard Science	Qualities	Metabolism	Soft Science	Systems
taxonomy	attribute	regulation	social relation	organism
science	drive	reproduction	profession	body
mathematics	folly	Krebs cycle	social science	hierarchy
pure mathematics	judgment	hypostasis	law	digestive system
applied mathematics	estimate	nutrition	politics	infrastructure
statistics	trait	growth	medicine	network
information theory	personality	anabolism	theology	system
computer science	character	bodily process	opinion	water system
information science	nature	catabolism	explanation	live body
information theory	thoughtfulness	gastrulation	anthropology	sensory system

3.3 Algorithm Spectral-2 The second algorithm we propose is a faster version of the algorithm in the previous section (Spectral-1). It uses a greedy strategy of recursive bisection to search over different values of k. Because of this strategy it need not find as high quality a clustering (as high a Q value) as the other approach, but it will be faster since in going from k clusters to $k+1$ only a portion of the data needs to be clustered rather than all of the data. The algorithm again takes as input the eigenvector matrix U_K as before and consists of the following steps:

1. Initialize k, the current number of clusters, to k_{min}. Initialize P, the best clustering assignment seen so far, to the clustering produced by running k-means with k set to k_{min} clusters. If $k_{min} = 1$, then simply initialize P to be one cluster containing all nodes in the graph.
2. Repeat until $k > K$ or no more splits are possible:
   (a) Set $P_{new} = P$
   (b) For each cluster V_c in P:
      i. If not already formed, form the matrix U_k from the first $k-1$ columns of U_K and scale the rows using the l^2-norm so they all have unit length
      ii. Form the matrix $U_{k,c}$ from U_k by keeping only rows corresponding to nodes in V_c
      iii. Run k-means with 2 clusters on $U_{k,c}$ to get two new sub-clusters,$V_{c,1}$ and $V_{c,2}$.
      iv. Form a new partition, P' by setting $P' = P$ and replacing the corresponding V_c with $V_{c,1}$ and $V_{c,2}$
      v. If $Q(P') > Q(P)$, accept the split by replacing the corresponding V_c in P_{new} with $V_{c,1}$ and $V_{c,2}$, otherwise reject it and leave P_{new} unchanged.
      vi. Assign k to be the (possibly new) number of clusters in P_{new}
   (c) Set $P = P_{new}$

The idea behind this algorithm is to start with k_{min} clusters and instead of rerunning k-means on the entire graph for subsequent values of k as we did in the previous algorithm, we instead try recursively splitting each cluster into two child clusters if the split produces a higher value of Q. By continuing this procedure until no more splits are possible or until K clusters have been found, we end up with a clustering with the highest value of Q encountered along the way. The particular algorithm above is order-sensitive in the sense that cycling through the clusters in a different order could produce different results—however, we have not noticed any particular sensitivity to order in the data sets described later in the paper. Unlike many other recursive bisection methods, model selection here is natural and straightforward. We choose to accept a split if the split results in a higher value of Q. Of course, the drawback with this algorithm is that we tradeoff speed with accuracy. The algorithm uses a greedy strategy, where a split can never be revoked and so one bad choice negatively affects all other choices further down the same branch. Thus, the quality of the results are not necessarily as good as with Spectral-1, although they are still generally competitive as we will see below. There is a subtlety in this algorithm in that, if left unchecked, each cluster for which a split attempt was made but failed would be retried again the next time through the loop. For this reason, we only allow a single attempt to split a given cluster.

3.4 Computational Complexity For the Spectral-1 algorithm, we run k-means K times, where in each case the dimensionality is $d = k - 1$. Standard k-means with a Euclidean distance metric has time complexity

$O(ndke)$ where n is the number of data points, d is the dimensionality of each point, and e is the number of iterations required for k-means to converge. Elkan [5] proposed a much faster version of k-means. It produces exactly the same results as standard k-means but uses various geometric inequalities to significantly reduce the number of distance computations required. Elkan found that with his proposed algorithm, the overall time complexity is roughly $O(nke)$ where e is the number of iterations. We use this algorithm for our implementation of k-means in both Spectral-1 and Spectral-2.

The resulting complexity for clustering in Spectral-1, using Elkan's fast k-means algorithm, is roughly $O(nK^2e)$. For Spectral-2 the computational complexity is not as easy to estimate. In the worst case (completely imbalanced clusters where the largest cluster is split at each iteration) it will have the same complexity as Spectral-1. However, in practice we have found that it is considerably faster than Spectral-1 and will show experimental results later in the paper that illustrate this.

In addition, for both algorithms there is the additional complexity of $O(mKh + nK^2h + K^3h)$ for computing the matrix of eigenvectors U_K, using the IRLM. Thus, Spectral-1 and Spectral-2 have an overall worst-case time complexity of $O(mKh+nK^2h+K^3h+nK^2e)$. Thus, for sparse graphs, where $m \sim n$, the algorithms will scale roughly linearly as a function of the number of nodes n. This is in contrast to Newman's algorithm which has complexity $O(n^2)$ even for sparse graphs, and thus does not scale up as well to large values of n.

4 Experimental Results

4.1 Clustering words from WordNet We first illustrate how different choices for the graph embedding can affect the quality of clustering. We use a relatively small unweighted graph extracted from the WordNet database of word meanings and semantic relationships [8]. The reason we chose this data set was because one can immediately judge the quality of the clusters since intuitively clusters should contain words that share common semantic features which are recognizable. We created an unweighted undirected graph where nodes represent words and an edge exists between two nodes if any of the following semantic relationships exist between them: synonymy, antonymy, hypernymy, hyponomy, meronymy, troponymy, causality, and entailment. The entire graph contains 82670 nodes. We extracted a subgraph comprised of all nodes whose shortest path distance away from the word "Science" is no more than 3. We also removed all nodes with degree 1 so that the graph layout would not be too cluttered. Adding this constraint had little effect on the quality of the clusters.

The resulting subgraph contains 230 nodes and 389 edges. Figure 2 shows the best clustering found by the Spectral-1 algorithm.[2] For this clustering there were 12 clusters and $Q = 0.696$. Table 1 shows ten representative words from five random clusters.

Figure 3 shows how the modularity Q varies with k when each of the following three types of matrices is used in step 1 of the first algorithm and the eigenvectors are computed exactly: standard Q Laplacian L_Q, normalized Q Laplacian $\mathcal{L}_Q$, and the transition matrix $D^{-1}W$. We can see that using the transition matrix in step 1 provides a very good approximation to the normalized Q matrix. We can also see that the standard Q Laplacian slightly underperforms both of these matrices. This agrees with observations by other authors that the normalized Laplacian gives better results than the standard Laplacian (e.g., [12],[14]).

4.2 Clustering American college football teams Our next example demonstrates the ability of both of our algorithms to identify known clusters. The unweighted network was drawn from the schedule of games played between 115 NCAA Division I-A American college football teams in the year 2000. Each node in this network represents a college football team and each edge represents the fact that two teams played together. Because the true conference to which each team belongs is known a-priori, and because inter-conference games are played more often than intra-conference games, the groups of teams that form conferences correspond to natural clusters that should be identifiable in the graph. Figure 4 shows that this is indeed the case where the Spectral-1 algorithm identified the correct number of clusters and, furthermore, each team assignment to a cluster made by the algorithm was correct.[3] Our second algorithm, Spectral-2, did almost as well making very few mistakes. The mistakes were:

1. North Texas was put into the Big 12 conference instead of Big West.
2. Arkansas State was put in Western Athletic instead of Big West.
3. EastCarolina was placed in Atlantic Coast instead of Conference USA
4. BigTen was split into two equal sized clusters: {Michigan, Ohio State, Wisconsin, Iowa, Illinois,

[2] Graphs are shown using the Fruchterman-Reingold layout.

[3] For the group of eight teams that do not belong to any conference, each was assigned to one of the conferences, e.g., Notre Dame, Navy, and Florida (Miami) to the Big East Conference.

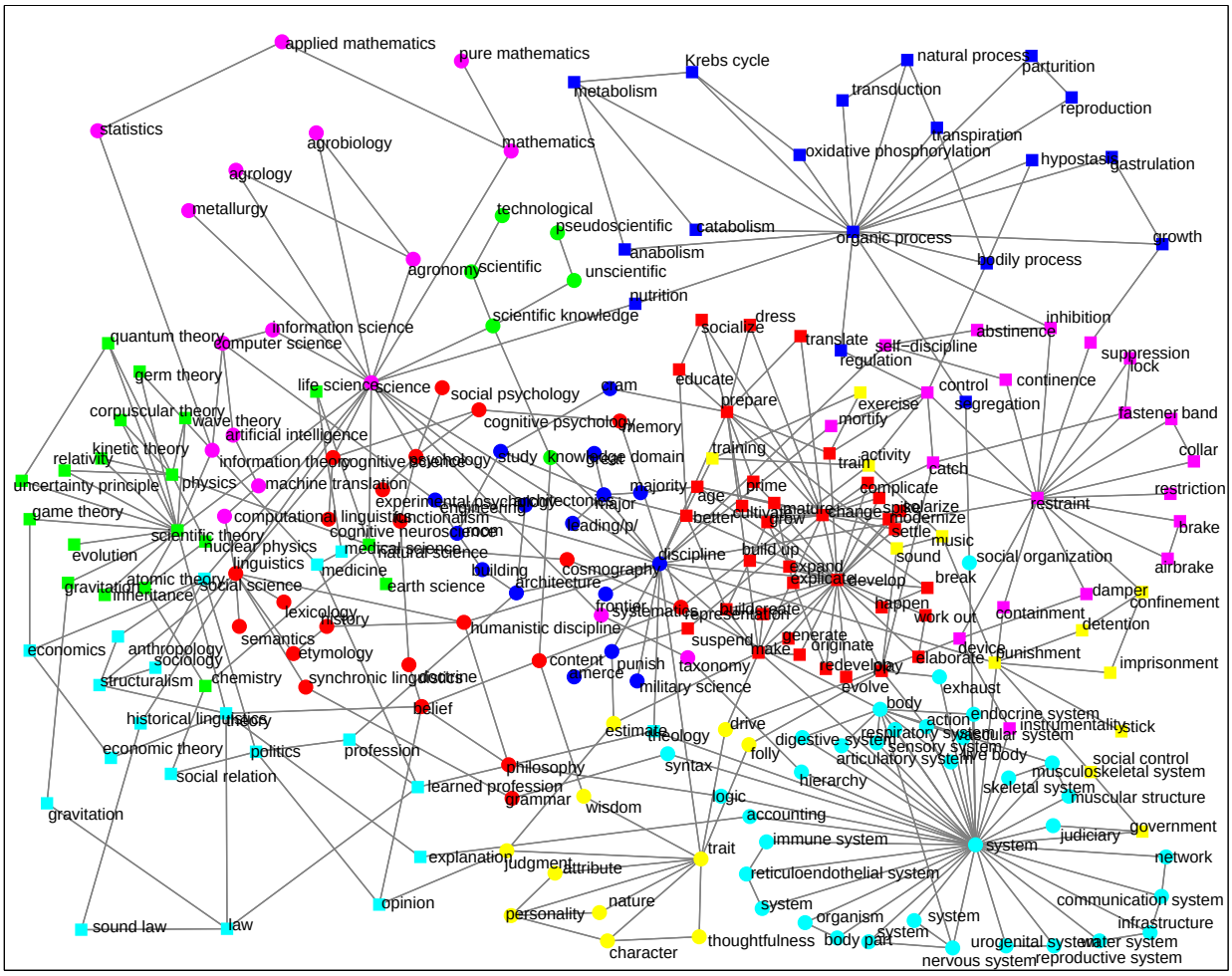

Figure 2: Clusters for WordNet data, $k = 12$ (best viewed in color).

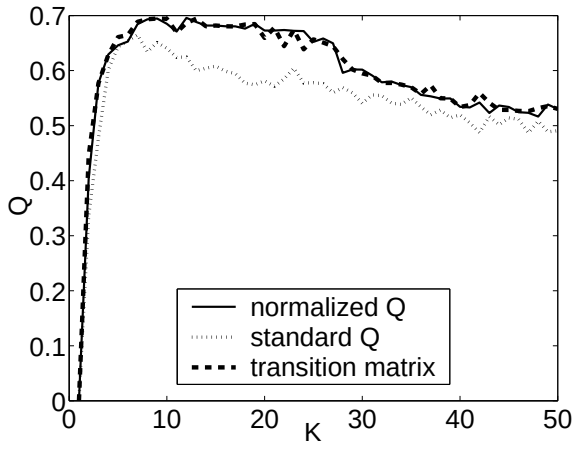

Figure 3: Q versus k for the WordNet data.

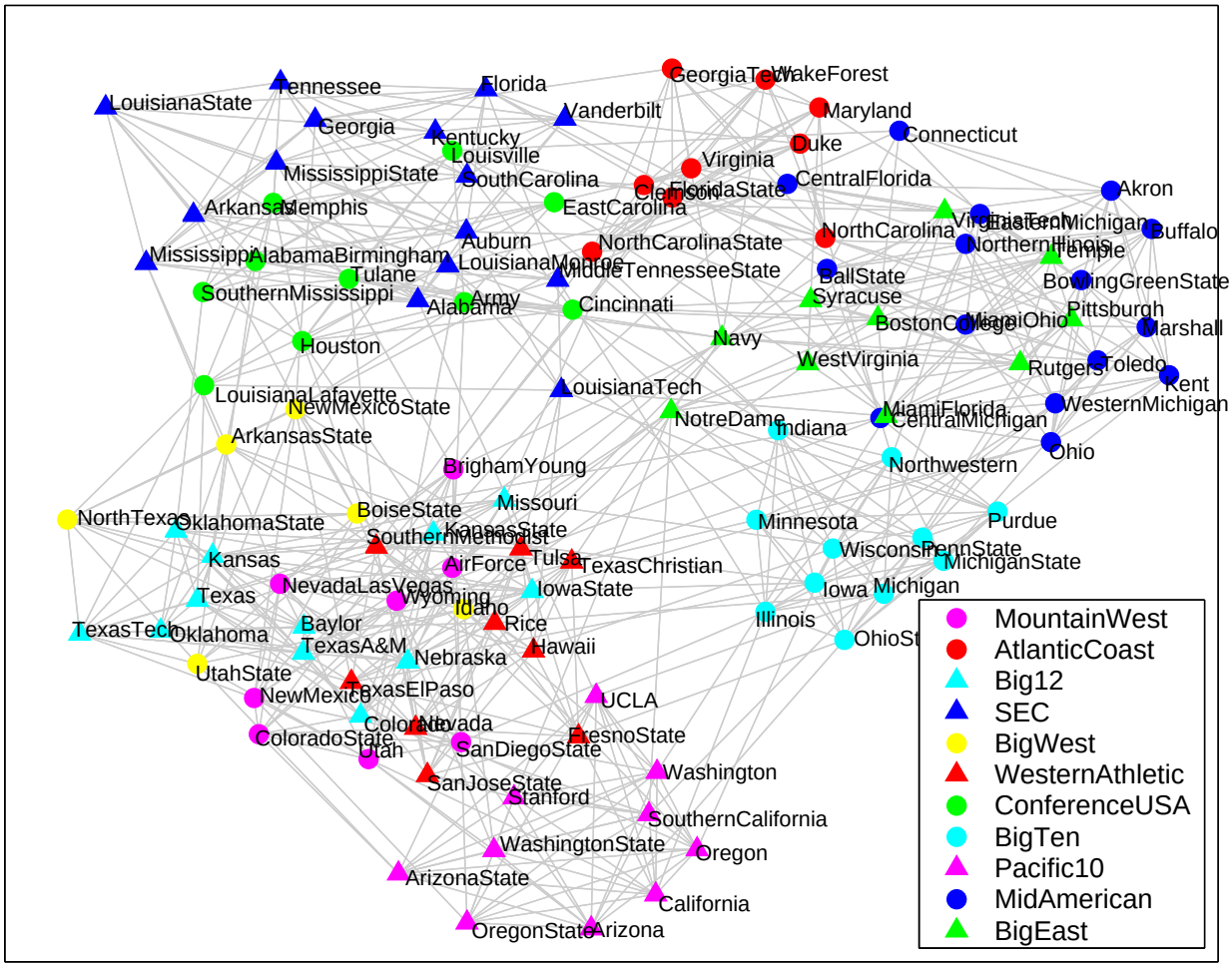

Figure 4: Clusters for NCAA Division I-A football teams, $k = 11$ (best viewed in color).

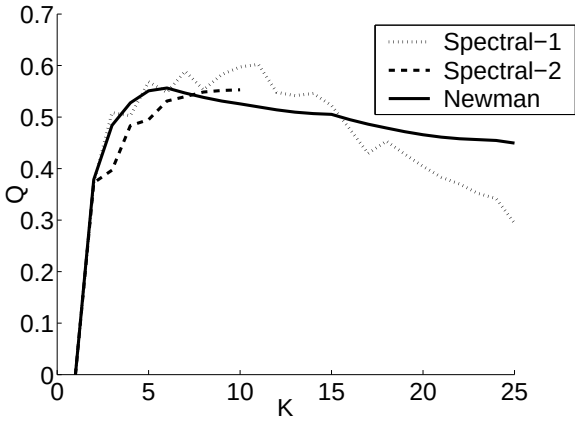

Figure 5: Q versus k for NCAA Division I-A football teams.

Michigan State} and {Northwestern, Purdue, Minnesota, Penn State, Indiana}

5. Mountain West, Pacific 10 and Big West were merged into one cluster

To summarize there were three individual mistakes where a single team was misplaced in the wrong conference, as well as one incorrect split and two splits that should have happened but did not.

Newman [10] proposed a hierarchical agglomerative clustering method for finding graph clusterings based on Q, with a worst-case time complexity of $O(n^2)$. At each step of the algorithm the two clusters are merged that result in the largest increase in Q, providing local iterative improvements to the overall Q score. The number of clusters k can be obtained from the resulting dendrograms by selecting the level of the tree for which $Q(\mathcal{P}_k)$ is highest. Figure 5 shows how the modularity Q varied with k for both of our algorithms as well as Newman's original greedy algorithm which uses the modularity Q as a distance measure to do agglomerative clustering. The peak for Spectral-1 was at k=11, Q=0.602, which corresponds precisely with the actual number of conferences in the NCAA Division I-A American college football league. We can see that Newman's algorithm underperforms this algorithm on this data set. The best clustering found by Newman's algorithm was for k=6, Q=0.556. As Newman [10] points out and we see in this example, his algorithm can miss the best clustering since it makes decisions purely at a local level whereas Q is inherently a global measure. The same is true for our faster, greedy algorithm. The best clustering found by Spectral-2 was k=10,Q=0.553. This is competitive with Newman's algorithm but as we will see in the timing experiments this algorithm runs significantly faster than Newman's algorithm.

4.3 Clustering authors publishing in NIPS For our final experiment we extracted a weighted coauthorship network from volumes 0-12 of the NIPS conference papers. The raw data contains 2037 authors and 1740 papers from which we created a coauthorship graph where nodes represent authors and edges represent coauthorship between the given pair of authors. We weighted each edge using the following weighting scheme: $w_{ij} = \sum_k \frac{1}{n_k - 1}$ where n_k is the number of coauthors for paper k, and w_{ij} represents the weight assigned to the edge between nodes i and j for which there was a coauthor relation. Building a coauthorship network from this data yields many disconnected components so we used only the dominant component which has 1061 nodes (authors) and 4160 edges (coauthorship pairs).

We ran both of our algorithms with K=100. Figure 6 shows the best clustering found by Spectral-1 where k=31 and Q=0.874. In this figure, the original coauthorship graph was shrunk so that now each node represents a cluster and the size of the node roughly indicates the size of each cluster. PageRank with Priors was used to label each cluster with the three authors whose importance was highest relative to the cluster to which they belong [13]. The resulting clusters clearly reflect various subcommunities of NIPS authors based on the first 12 years of the conference. Figure 7 shows how the modularity Q varies with k. We found that on this data set Newman's algorithm marginally outperformed our Spectral-1 algorithm, although both algorithms were very close in terms of the optimal value of Q found: Q=0.874 and k=31 for Spectral-1 vs. Q=0.876 and k=33 for Newman's algorithm. Spectral-2 did not do as well although it still gave competitive results finding a clustering for k=44 and Q=0.861.

This brings up an important point about the difference between Newman's algorithm and our faster, greedy Spectral-2 algorithm. Newman's algorithm starts with n clusters, one for each node in the graph, and continues to merge clusters one at a time whereas our faster algorithm starts with one cluster, all nodes in the graph being assigned to it, and continues to split clusters one at a time. Thus, as we saw in the previous example with the college football data set, Newman's algorithm made some mistakes early on in the merging process which caused Q to reach a maximum only after k was already much smaller than the correct solution for $k = 11$. Our faster algorithm, Spectral-2, also made mistakes early on, in the splitting process, causing it to overshoot the more optimal values of Q found by the other two algorithms in the vicinity of $32 \leq k \leq 33$ and instead find a maximum value of Q for much larger k, in this case $k = 44$. Although both algorithms are able to find clusterings that are quite competitive with Spectral-1, they both potentially suffer from the problem of overshooting, although each in opposite directions, because of the potential limitations of the greedy strategy. Nevertheless, this example also highlights the fact that there are many graphs for which a greedy strategy can perform quite well (as also documented by Newman [10]).

4.4 Timing Experiments In this section, we present timing results for each of the experiments conducted in the experiments just described. In addition, we reran Spectral-1 and Spectral-2 for $K = 25$ on the "Science" network and for $K = 50$ on the NIPS coauthorship network to highlight how the choice of K affects performance. All algorithms were implemented in

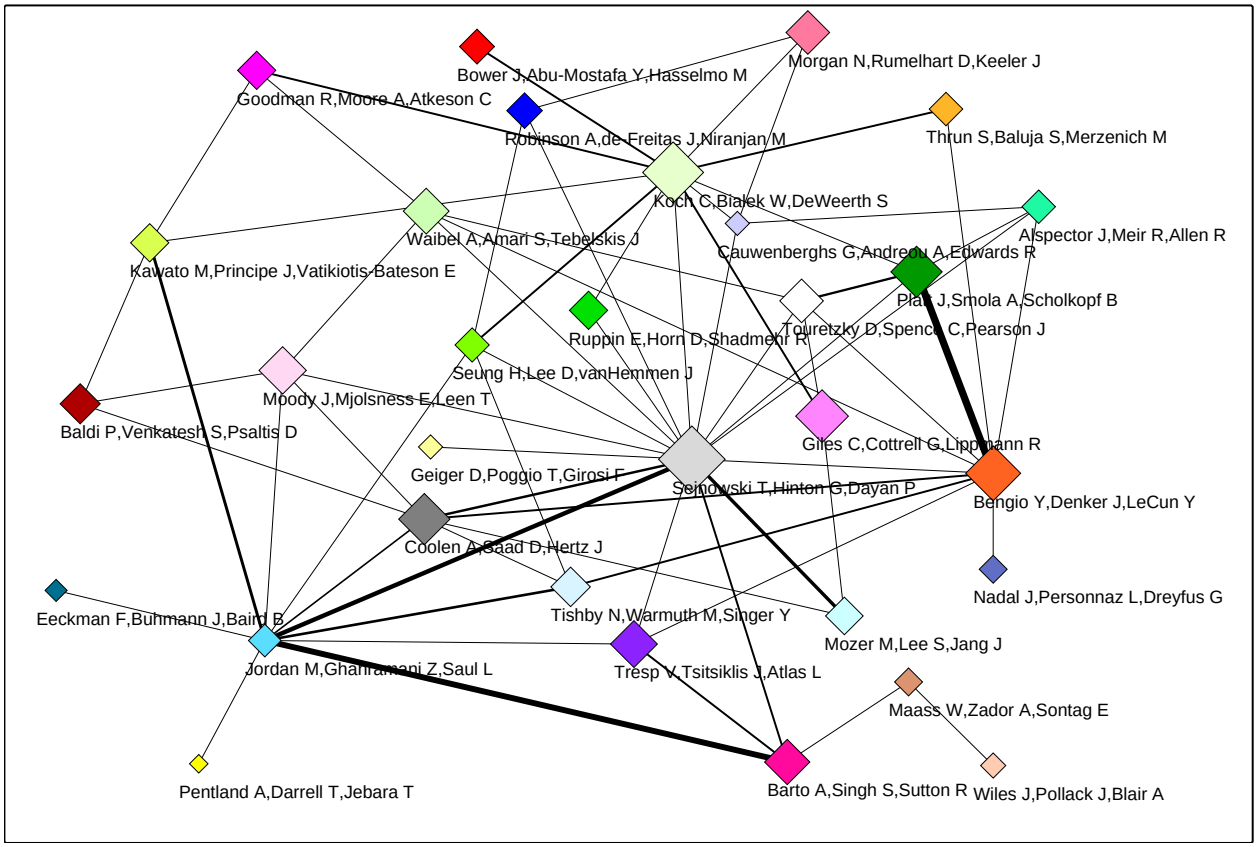

Figure 6: Clustering for NIPS co-authorship data, $k = 31$ (best viewed in color).

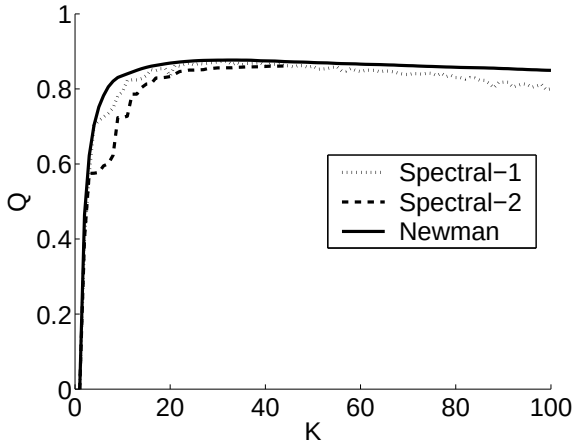

Figure 7: Q versus k for NIPS coauthorship data.

Matlab and run on a 1.2 GHz Pentium II laptop. We used the sparse eigendecomposition routine *eigs* in Matlab to compute the eigenvectors using the IRLM.

Table 2: Timing results for separate components of Spectral-1 in seconds.

Name	n	K	*eigs*	Clustering
Football	115	25	0.16	2.29
Science	230	25	0.38	3.43
Science	230	50	0.67	12.36
NIPS	1061	50	6.53	40.41
NIPS	1061	100	15.49	292.15

Table 2 shows the timings for each of the key components of the Spectral-1 algorithm. Clustering takes most of the time, especially as K and n increase.

Table 3: Timing results for separate components of Spectral-2 in seconds.

Name	n	K	*eigs*	Clustering
Football	115	25	0.16	0.15
Science	230	25	0.38	0.23
Science	230	50	0.67	0.30
NIPS	1061	50	6.53	1.33
NIPS	1061	100	15.49	2.01

Table 3 shows the timings for each of the key components of the Spectral-2 algorithm. For Spectral-2, as K increases, running *eigs* becomes an increasingly large performance bottleneck. The time taken for clustering is low since we never have to re-run the k-means algorithm on the entire graph. This includes the time taken to determine whether or not to accept a split which involves computing Q which takes little time since we don't recompute Q from scratch but instead update Q based on the edges that have been reassigned.

Table 4: Overall timing results in seconds.

Name	n	K	Spec-2	Spec-1	Newman
Football	115	25	0.41	3.11	7.74
Science	230	25	0.77	4.23	8.38
Science	230	50	1.06	13.96	8.38
NIPS	1061	50	10.57	51.57	387.15
NIPS	1061	100	22.14	321.14	387.15

Table 4 shows the overall times (in seconds) for each of the five experiments.[4] Perhaps most interesting is to observe how the interaction between K and n affects the results. Spectral-1 is generally faster than Newman's algorithm although for large enough values of K and small enough values of n, Newman's algorithm can be faster. Spectral-2 is generally at least an order of magnitude faster than the other two algorithms although as K gets larger the difference in speed is less pronounced.

5 Discussion

The idea of reducing a combinatorial graph partitioning problem to a geometric vector space partitioning problem using spectral techniques is by no means new. Some of the earliest breakthroughs can be attributed to Hall [7] and Fiedler [6]. Alpert and Yao [1] showed that when the full eigenspace is used, certain graph partioning problems exactly reduce to vector partioning ones. More recently, Brand and Huang [3] presented theoretical results precisely characterizing how compacting the eigenbasis is able to magnify structure in the data. Furthermore, Chung [4] and others have laid much of the foundational work in spectral graph theory, on which a large part of the subsequent theoretical analysis of spectral clustering methods is based.

The key idea in this paper is to reverse engineer Newman's Q function into a spectral framework in which any input graph can be optimally embedded into Euclidean space. Once the input graph is represented in a Euclidean space, we can then use fast geometric clustering algorithms such as k-means to identify the clusters. Any algorithmic framework developed in this way faces a large search problem since both k (the number of clusters) and the dimensionality of the embedding which maximize Q need to be explored. Both algorithms for fixed k choose the dimensionality of the embedding to be k-1 (e.g. for k=2, we just use the top eigenvector). The assumption here is that while it may be possible, in some cases, to use fewer dimensions and still find a good clustering for fixed k, while also making the algorithm even faster, it is better to be conservative. Experiments have shown that the higher the dimensionality (the more eigenvectors), the better the clusterings produced, although we did not find that having the dimensionality of the embedding exceed k-1 helped in any way.

Both algorithms empirically track the performance of Newman's algorithm quite closely. The slower, more

[4] Note that the times for Spectral-1 and Spectral-2 are slightly larger than the sums of the corresponding times in Tables 2 and 3 due to additional overhead in the algorithms.

accurate algorithm (Spectral-1) can produce higher-quality clusterings than Newman's because of the non-greedy search heuristic. The Spectral-2 algorithm could be viewed as a top-down divisive search alternative to Newman's bottom-up agglomerative search, in a general hierarchical clustering context, with attendant advantages and disadvantages to each in terms of the greedy search strategy, as well as having significant differences in their computational characteristics.

Other search heuristics approaches are also possible and may lead to different trade-offs between cluster quality and computation time. For example, combining both of our algorithms into a hybrid algorithm may yield a fruitful trade-off between speed and cluster quality. For graphs where the number of clusters to search over is large, Newman's hierarchical clustering approach may be the preferred method given that it operates directly on the graph without any need for embedding the graph into a Euclidean vector space. Our algorithms, in contrast, use sparse eigenvector techniques which scale quadratically with the number of clusters to search over. However, when the number of clusters to search over is small, and n the number of nodes increases, the $O(n^2)$ complexity of hierarchical clustering can quickly become intractable. In contrast, the two algorithms we propose here will scale relatively well to large graphs.

6 Conclusions

In this paper we have shown how the recently proposed Q function can be used to find high quality graph clusterings. We give a precise analytical expression which when maximized returns a discrete assignment matrix X that represents the optimal partitioning of a graph according to the Q function for fixed k. Because maximizing this expression is NP-Complete, we show how the discrete maximization can be approximated as a continuous one that is easily solvable by performing eigenvector decomposition on a matrix L_Q, which we call the Q-Laplacian. We present two algorithms which attempt to search over different values of k to find the best value of k and the accompanying best clustering. The first algorithm we present searches independently for a best clustering for each value of k. Unlike Newman's algorithm, which optimizes Q by local iterative improvement, this algorithm seeks a direct global maximum of Q. The second algorithm we present is similar to Newman's algorithm in that it uses a local greedy search heuristic; however, it is based on a top-down strategy of splitting clusters that lead to higher values of Q and is thus much faster than the other two algorithms for $K \ll n$. Empirical results suggest that both methods provide high quality graph clusterings on a variety of graphs that exhibit community structure, and both methods scale linearly in the number of edges, allowing for applications to large sparse graphs.

Acknowledgements The data used were generously made available by the Cognitive Science Laboratory at Princeton University (WordNet), Mark Newman (College Football) and Sam Roweis (NIPS).

References

[1] C. Alpert and S. Yao, Spectral partitioning: the more eigenvectors the better. In *Proceedings of 32nd ACM/IEEE Design Automation Conference*, 1995, pp. 195-200.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. Vorst, eds., Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, SIAM, Philadelphia, 2000.

[3] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. *9th International Conference on Artificial Intelligence and Statistics*, 2002.

[4] F. Chung. Spectral graph theory. Number 92 in *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

[5] C. Elkan. Using the triangle inequality to accelerate k-Means. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 147-153.

[6] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23 (1973), pp. 298-305.

[7] K. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 11(3)(1970), pp. 219-229.

[8] G. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3 (1990), pp. 235-312.

[9] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, 026113 (2004).

[10] M. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69, 066133 (2004).

[11] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2002, pp. 849-856.

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (2000), pp. 888-905.

[13] S. White and P. Smyth. Algorithms for discovering relative importance in graphs. In *Proceedings of Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 266-275.

[14] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings OF IEEE International Conference on Computer Vision*, 1999, pp. 975-982.