# An Efficient Spectral Algorithm for Network Community Discovery and Its Applications to Biological and Social Networks

Jianhua Ruan and Weixiong Zhang
Department of Computer Science and Engineering
Washington University in St Loius
One Brookings Dr, St Louis, MO 63130
{jruan,zhang}@cse.wustl.edu

## Abstract

*Automatic discovery of community structures in complex networks is a fundamental task in many disciplines, including social science, engineering, and biology. Recently, a quantitative measure called modularity (Q) has been proposed to effectively assess the quality of community structures. Several community discovery algorithms have since been developed based on the optimization of Q. However, this optimization problem is NP-hard, and the existing algorithms have a low accuracy or are computationally expensive. In this paper, we present an efficient spectral algorithm for modularity optimization. When tested on a large number of synthetic or real-world networks, and compared to the existing algorithms, our method is efficient and and has a high accuracy. We demonstrate our algorithm on three applications in biology, medicine, and social science. In the first application, we analyze the communities in a gene network, and show that genes in the same community usually have very similar functions, which enables us to predict functions for some new genes. Second, we apply the algorithm to group tumor samples based on gene expression microarray data. Remarkably, our algorithm can automatically detect different types of tumor without any prior knowledge, and by combining our results and clinical information, we can predict the outcomes of chemotherapies with a high accuracy. Finally, we analyze a social network of Usenet newsgroup users, and show that, without any semantic information, we can discover the organization of the newsgroups, and detect users groups with similar interests.*

## 1 Introduction

The study of complex networks has become a fast growing subject in many disciplines, including physics, biology, and social science. At least part of the reason can be attributed to the discovery that real-world networks from totally different sources can share surprisingly high similarities in their topological properties, such as the power-law degree distributions and high clustering coefficients. (See [1, 20] for reviews.)

One of the key properties in complex networks that have attracted a great deal of interest recently is the so-called community structures, i.e. relatively densely connected sub-networks [21]. Community structures have been found in social and biological networks, as well as technological networks such as the Internet and power grid. Automatically discovering such structures is fundamentally important for understanding the relationships between network structures and functions, and has many practical applications. For example, identifying communities from a collaboration network may reveal scientific activities as well as evolution and development of research areas [15]; detecting hidden communities on the World Wide Web may help prevent crime and terrorism [4]; discovering communities in a network of biomolecules may help us better understand the organizational principles of the cell, and identify important biological pathways for further studies [29, 34].

Community discovery is similar but not equivalent to the conventional graph partitioning problem, both of which are amount to clustering vertices into groups. A key challenge for the former, however, is that the algorithm has to decide what is the "best", or in other words, the "most natural" partition of a network. In contrast, in conventional graph partitioning, the user has to provide information such as the number of partitions or the sizes of the sub-networks. Furthermore, a community discovery algorithm should not force a network to be partitioned if there is no good community structures, while it is always possible for a graph partitioning algorithm to return some arbitrary results.

To design effective community discovery algorithms, Newman and Girvan [22] proposed a quantitative measure,

called modularity ($Q$), to assess the quality of community structures, and formulated community discovery as a optimization problem. Since optimizing $Q$ is a NP-hard problem, several heuristic methods have been developed, as surveyed in [6]. The fastest algorithm available uses a greedy strategy and suffers from poor quality [5]. A more accurate method is based on simulated annealing, which requires a prohibitively long running time on large networks [14]. Several spectral algorithms have been developed, which have relatively good performance, but still inefficient for large networks [33, 21].

In this paper, we propose a spectral algorithm that is effective in finding high quality communities as well as efficient on large networks. The algorithm adopts a recursive strategy to partition networks while optimizing $Q$. Unlike the existing algorithms, our method is a hybrid of direct $k$-way partitioning and recursive 2-way partitioning strategies [33, 21]. We evaluate our algorithm on a large number of synthetic and real-world networks. The results show that the algorithm is more efficient and more accurate than a recursive 2-way partitioning method. Compared to a direct $k$-way partitioning method, our algorithm is much more efficient, while having a comparable accuracy.

We demonstrate our algorithm on several real applications in different domains. First, we apply our algorithm to identify functional modules from a network of genes in the yeast *S. cerevisiae*, and predict functions for some genes based on the community structures. Second, we use the algorithm to classify tumor samples based on gene expression microarray data, which is a challenging and important task for effective therapy of cancer. Finally, we analyze the community structure in a social network of Usenet newsgroup users. We show that, by simply looking at the communication history among the users, our algorithm can identify groups of users sharing common interests, without any semantic information of the messages.

The paper is organized as follows. In Section 2, we introduce some basic concepts, notations, and previous works. We describe our algorithm and its complexity in Section 3, and present some evaluation results in Section 4. Finally, we demonstrate our algorithm in several real applications in Section 5, and conclude in Section 6.

## 2 Preliminaries

### 2.1 Spectral graph partitioning

Let $G = (V, E)$ be a network of $n$ vertices in $V$ and $m$ edges in $E$. Let $A = (A_{ij})$ be the adjacency matrix of $G$. A graph partitioning problem is to find two or more vertex subsets of nearly equal sizes, while minimizing the number of edges cut by the partitioning [11]. Known to be NP-hard, the problem exists in many real applications, such as circuit design and load balancing in distributed computing. Many heuristic methods have been developed for the problem, among which spectral methods have received much attention and are the most popular.

Spectral graph partitioning is in fact a family of methods. These methods depend on the eigenvectors of the Laplacian matrix or its relatives of a graph. Depending on the way they partition a graph, spectral methods can be classified into two classes. The first class uses the leading eigenvector of a graph Laplacian to bi-partition the graph. The second class of approaches computes a $k$-way partitioning of a graph using multiple eigenvectors. We briefly review some representatives of these two classes of algorithms below.

Let $D$ be the diagonal degree matrix of $A$, i.e. $D_{ii} = \sum_j A_{ij}$. $L = D - A$ is called the Laplacian matrix of $G$. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues and $\mu_1, \mu_2, \cdots, \mu_n$ be the corresponding eigenvectors for the generalized eigenvalue problem $L\mu = \lambda D\mu$. It can be shown that $\lambda_1 = 0$, and $\mu_1 = \mathbf{1}$, a vector with all ones.

Given the above notation, a representative of bi-partitioning, the *SM* algorithm [26], works as follows. (1) Compute $\mu_2$, the second smallest generalized eigenvector of $L$. (2) Conduct a linear search on $\mu_2$ to find a partition of the graph to minimize a normalized cut criterion [26]. It has been shown that when certain constraints are satisfied, the *SM* algorithm can reach the optima of normalized cuts [26]. To find more than two clusters, the *SM* algorithm can be applied recursively.

The most popular algorithm in the second class, the *NJW* algorithm [23], finds a $k$-way partition of a network directly as follows, where $k$ is given by the user. (1) Compute the $k$ smallest generalized eigenvectors of $L$ and stack them in columns to form a matrix $Y = [\mu_1, \mu_2, \cdots, \mu_k]$. (2) Normalize each row of $Y$ to have unit length. (3) Treat each row as a point in $R^k$, and apply standard $k$-means algorithm (or any other geometric clustering algorithm) to group them into $k$ clusters.

Note that the above algorithm is slightly different from but equivalent to the original *NJW* algorithm. The original algorithm computes the $k$ *largest* eigenvectors of a normalized matrix $N = D^{-1/2}AD^{-1/2}$. It can be shown that, if $\lambda$ and $\mu$ are solutions to $L\mu = \lambda D\mu$, $1 - \lambda$ and $D^{1/2}\mu$ are precisely the eigenvalues and eigenvectors of $N$ [32]. Therefore, $D^{1/2}\mu_i, i = 1, \cdots, k$, are the largest $k$ eigenvectors of $N$. Since $D$ is diagonal, $Y$ obtained from $N$ is the same as $Y$ obtained from $L$, except that each row in the former is scaled by a factor of $\sqrt{D_{ii}}$. The scaling factor disappears after normalizing each row to have unit length. This small modification puts *NJW* in the same framework as the *SM* algorithm, and is numerically more stable, since no division of $D$ is involved.

### 2.2 Modularity and community structures

Given a partition of a network, $\Gamma^k$, which divides its vertices into $k$ communities, the modularity is defined as $Q(\Gamma^k) = \sum_{i=1}^{k}(e_{ii}/c - (a_i/c)^2)$, where $e_{ii}$ is the number of edges with both vertices within community $i$, $a_i$ is the number of edges with one or both vertices in community $i$, and $c$ is the total number of edges [22]. Therefore, the $Q$ function measures the fraction of edges falling within communities, subtracted by what one would expect if the edges were randomly placed. A larger $Q$ value means stronger community structures. If a partition gives no more within-community edges than would be expected by chance, the modularity $Q \leq 0$. For a trivial partitioning with a single community, $Q = 0$. It has been observed that most real-world networks have $Q > 0.3$ [22].

The $Q$ function provides a good quality measure to compare different community structures. Several algorithms have been developed to search for community structures by looking for the division of a network that optimizes $Q$ (see [6] for a survey). White and Smyth proposed a spectral algorithm (*WS*), which is effective on small networks [33]. They show that, when the number of communities $k$ is given, the optimization of $Q$ is equivalent to an eigen decomposition problem, if relaxing the discrete membership constraint [33]. Therefore, they directly applied a $k$-way spectral graph partitioning algorithm for this purpose. To automatically determine the number of communities, the spectral algorithm is executed multiple times, with $k$ ranging from the user defined minimum $K_{min}$ to maximum $K_{max}$ number of communities. The $k$ that gives the highest $Q$ value is deemed the most appropriate number of communities. A slightly modified version of the *WS* algorithm is as follows. (1) For each $k$, $K_{min} \leq k \leq K_{max}$, apply *NJW* to find a $k$-way partition, denoted as $\Gamma_k$. (2) $k^* = \arg\max_k Q(\Gamma_k)$ is the number of communities, and $\Gamma^* = \Gamma_{k^*}$ is the best community structure.

The key difference between the original *WS* algorithm [33] and the one above is the underlying spectral algorithm. The original algorithm uses the second to the $k$-th *largest* eigenvectors of a transition matrix $P = D^{-1}A$, and ignores the first eigenvector. It can be shown that the eigenvalues and eigenvectors of $P$ are precisely $1 - \lambda$ and $\mu$, where $\lambda$ and $\mu$ are the solutions to the generalized eigenvalue problem $L\mu = \lambda D\mu$ [32]. Therefore, the underlying spectral algorithm in the original *WS* algorithm is equivalent to *NJW* except that the first eigenvector. The authors ignored the first eigen vector since it is a constant, $\mathbf{1}$. However, in the *NJW* algorithm, a normalization step is applied so that each row of $Y$ has unit length. After the normalization, the first column of $Y$ is often not constant. We noticed that including the first column often gives higher $Q$ values. Therefore, we use *NJW* as the underlying spectral algorithm.

While the *WS* algorithm is effective in finding good community structures, it scales poorly to large networks, because it needs to execute $k$-means up to $K_{max}$ times. Without any prior knowledge of a network, one may overestimate $K_{max}$ in order to reach the optimal $Q$. For sparse networks, $K_{max}$ can be linear in the number of vertices in the worst case, making it impractical to iterate over all possible $k$'s for large networks.

## 3 The *Kcut* algorithm

In order to develop a method that scales well to large networks while retaining effectiveness in finding good communities, we may take the strategy used in the *SM* algorithm, i.e., to recursively divide a network into smaller ones. However, two issues remain. First, when should the algorithm halt, or in other words, how do we decide whether a (sub)network should be partitioned? Since our goal is to find a partition with a high modularity, we can test whether the $Q$ value increases after the partition. If no partition can improve the modularity, the (sub)network should not be divided. Second, it has been empirically observed that if there are multiple communities, using multiple eigenvectors to directly compute a $k$-way partition is better than recursive bi-partitioning methods [23]. Here, we propose an algorithm that is a unique combination of recursive partitioning and direct $k$-way methods, which will achieve the efficiency of a recursive approach, while also having the same accuracy as a direct $k$-way method.

We follow a greedy strategy to recursively partition a network to optimize $Q$. Unlike the existing algorithms that always seek a bi-partition, we adopt a direct $k$-way partition strategy as in the *WS* algorithm. Briefly, we compute the best $k$-way partition with $k = 2, 3, \cdots, l$ using the NJW algorithm, and select the $k$ that gives the highest $Q$ value. Then for each subnetwork the algorithm is recursively applied. To reduce the computation cost, we restrict $l$ to small integers. As we will shown in experiments, the algorithm with $l$ as small as 3 or 4 can significantly improve modularity over the standard bi-partitioning strategy. Furthermore, the computation cost is also reduced with a slightly increased value of $l$ compared to bi-partitioning.

Given a network $G$ and a small integer $l$ that is the maximal number of partitions to be considered for each subnetwork, our algorithm *Kcut* executes the following steps.

1. Initialize $\Gamma$ to be a single cluster with all vertices, and set $Q = 0$.
2. For each cluster $P$ in $\Gamma$,
   (a) Let $g$ be the subnetwork of $G$ containing the vertices in $P$.
   (b) For each integer $k$ from 2 to $l$,

i. Apply *NJW* to find a $k$-way partitioning of $g$, denoted by $\Gamma_k^g$,

ii. Compute new $Q$ value of the network as $Q'_k = Q(\Gamma \bigcup \Gamma_k^g \setminus P)$.

(c) Find the $k$ that gives the best $Q$ value, i.e., $k^* = \arg\max_k Q'_k$.

(d) If $Q'_{k^*} > Q$, accept the partition by replacing $P$ with $\Gamma_{k^*}^g$, i.e., $\Gamma = \Gamma \bigcup \Gamma_{k^*}^g \setminus P$, and set $Q = Q'_{k^*}$.

(e) Advance to the next cluster in $\Gamma$, if there is any.

The inner loop, step 2(b), is similar to the first step of the *WS* algorithm, except that in 2(b)(ii) we compute the modularity of the whole network $G$, which is different from the modularity $Q(\Gamma_k^g)$. On the other hand, we do not need to iterate over all communities in the network to re-compute $Q$. From the definition of $Q$ in Section 2.2, the contribution of each community towards $Q$ is independent of the other communities. Therefore, after $g$ is partitioned, $Q$ can be efficiently updated with the communities that have just been created in $g$. At step 2(c), we decide the best way to partition $g$ that can improve $Q$ the most. This step turns out to be crucial in identifying globally good community structures with high $Q$ values. At step 2(d), we test if partitioning $g$ can contribute positively towards $Q$, and the partition is accepted only if $Q$ increases. When the algorithm terminates, no communities can be further created to improve $Q$, thus $\Gamma$ contains the best community structure.

### 3.1 Computational complexity

We first review the computational complexity of the *WS* algorithm, since the inner loop of *Kcut* is simply the *WS* algorithm, except that the computation of $Q$ is slightly different. The *WS* algorithm contains two major components: computing eigenvectors and executing $k$-means to partition the network. Note that although *WS* calls *NJW* multiple times, the eigen problem needs to be solved only once to obtain all $K_{max}$ eigenvectors. To compute eigenvectors, we used the *eigs* function in MATLAB, which has a time complexity in $O(mKh + nK^2h + K^3h)$, where $m$ and $n$ are respectively the numbers of edges and vertices of the graph, $K = K_{max}$ is the number of eigenvectors to be computed, and $h$ is the number of iterations for *eigs* to converge [33]. Since $K < n$, the running time of $eigs$ can be simplified to $O(mKh + nK^2h)$. Second, we adopted a fast $k$-means algorithm [8] in our implementation, which takes approximately $O(nKe)$ time, where $e$ is the number of iterations for $k$-means to converge. Since $k$-means is called $K$ times, the total running time is $O(mKh + nK^2h + nK^2e)$, where the first two terms are for $eigs$ and the last term is for $k$-means. Assuming $e$ and $h$ constants, the overall time complexity of *WS* is $O(mK + nK^2)$, which can be close to $O(n^3)$, since the maximal number of communities for a sparse network may be linear in $n$.

The running time of *Kcut* depends on the depth of the recursive calls. In the worst case, the partitions can be highly imbalanced, and the depth of the recursion is merely the number of partitions produced, $K$. A more practical estimate, however, is the average depth, which is close to $\log_l K$, where $l$ is the maximal number of partitions considered by *NJW*. Therefore, the running time taken by *eigs* can be estimated to be $O((mlh + nl^2h)\log_l K)$, which can be further simplified to $O(mlh \log_l K)$, since $l$ is small and therefore in general $m > nl$. Similarly, the average-case running time taken by $k$-means is $O(nl^2e \log_l K)$, and the total complexity is given by $O((mlh + nl^2e)\log_l K)$.

Our experimental results show that for large networks and small values of $l$, the time taken by $eigs$ dominates, giving an overall time complexity in $O(mlh \log_l K) = O(mh \ln K \frac{l}{\ln l})$ for *Kcut*. Therefore, assuming $h$ is a constant, also given that $l$ is small and $K = O(n)$, the total complexity is $O(m \log n)$, which is much smaller than the $O(n^3)$ running time of the *WS* algorithm. An important observation from the analysis is that the total running time of *Kcut* is not a monotonically increasing function of $l$. Analytically, the minimum value of $l/\ln l$ is achieved at $l = 3$. Empirically, we observed that *Kcut* is most efficient with $l = 3$ to 5 (see Section 4.2).

The memory complexity of both algorithms is $O(m)$, linear to the number of edges.

### 3.2 Related methods

Besides our algorithm and the *WS* method, several other algorithms have also been developed for identifying communities by modularity optimization. Newman proposed an algorithm that is based on recursive spectral bi-partitioning [21]. The algorithm computes the leading eigenvector of a so-called modularity matrix, and divides the vertices into two groups according to the signs of the elements in the eigenvector. The algorithm runs recursively on each subnetwork, until no improvement to $Q$ is possible. Compared to our method, this algorithm is faster for small networks, since no $k$-means is performed. On the other hand, the modularity matrix is very dense, with almost no zero entries. Therefore, the algorithm takes $O(n^2)$ memory even for sparse networks, in contrast to $O(m)$ for our method. Furthermore, the algorithm takes $O(n^2 \log n)$ running time, therefore, it does not scale well to large networks. Importantly, we will show that by combining $k$-way partitioning with a recursive method, *Kcut* usually achieves higher modularity than the Newman method.

White and Smyth also proposed a fast greedy algorithm, *spectral-2*, in addition to their optimal *WS* algorithm [33]. Like the Newman method, *spectral-2* is recursive and uses

spectral bi-partitioning to optimize $Q$. In this algorithm, all $K_{max}$ (a user provided parameter) leading eigenvectors of a transition matrix $P$ (see section 2.2) are computed and used throughout the subsequent recursive partitions. For large networks, however, it is difficult to estimate the number of communities in advance, and expensive to compute a large number of eigenvectors.

There are also several methods that are not spectral-based. The edge betweenness algorithm [12] and the extremal optimization algorithm [7] are known to be very slow, with $O(n^3)$ and $O(n^2 \log^2 n)$ running time, respectively. Another greedy approach, the *C*NM algorithm [5], has approximately the same time complexity ($O(m \log^2 n)$) as our method, but the communities returned often have poor quality [21].

# 4 Evaluation

We now evaluate our algorithm on a variety of networks and compare it with three existing algorithms that were mentioned in Section 3.2: the *WS* algorithm, the *CNM* algorithm, and the Newman's algorithm (*NM*). In what follows, the results of our algorithm are denoted by *K-2*, *K-3*, $\cdots$, for $l = 2, 3, \cdots$. Note that Newman suggested in [21] a refining step to improve $Q$ after the initial partitioning. To make a fair comparison, this refining step was omitted in our study, since in theory the same strategy can be applied to any other algorithm as well.

## 4.1 Computer-generated networks

To evaluate *Kcut*, we first tested it on computer-generated networks with artificially embedded community structures. Each network had 256 vertices forming 8 communities of equal sizes. Edges were randomly placed with probability $p_{in}$ between vertices within the same community and with probability $p_{out}$ between vertices in different communities. We varied $p_{in}$ from 0.8 to 0.3, representing networks with dense to sparse communities. For each $p_{in}$, we varied $p_{out}$ from 0 to $\frac{p_{in}}{10}$ with an increment of $\frac{p_{in}}{50}$. For each pair of ($p_{in}$, $p_{out}$), we generated 100 networks and clustered them with *WS* ($K_{min} = 2$, $K_{max} = 15$), *Kcut* ($l = 2, 3, 4$ and 5), and *NM* algorithms. To measure the accuracy of the results, we computed the Jaccard Index [30], which is roughly the percentage of within-community edges that were predicted correctly. The Jaccard Index between the true community structure ($\Gamma$) and predicted community structure ($\Gamma'$) is defined as

$$J(\Gamma, \Gamma') = \frac{|S(\Gamma) \cap S(\Gamma')|}{|S(\Gamma) \cup S(\Gamma')|}, \quad (1)$$

where $S(\Gamma)$ and $S(\Gamma')$ are the sets of within-community vertex pairs in $\Gamma$ and $\Gamma'$, respectively.

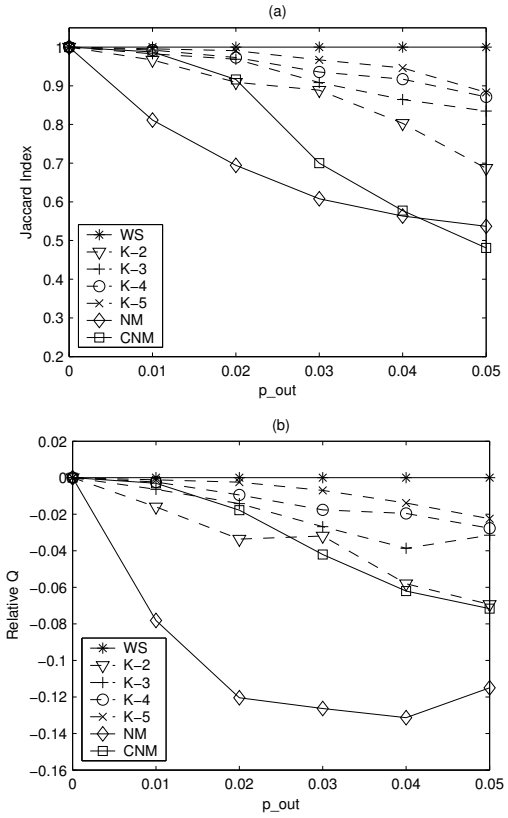


**Figure 1. Results on computer-generated networks.** $Q_{relative} = Q_{discovered} - Q_{true}$.

Fig. 1(a) shows the Jaccard Index as a function of $p_{out}$ for $p_{in} = 0.5$. Results for other values of $p_{in}$ or using other types of accuracy measurement are similar (data not shown). The *WS* algorithm, which explicitly searches over all $k$'s, has the best accuracy. On the other hand, *Kcut* with large $l$ values can better approximate *WS* than with small $l$ values. Moreover, as shown in Fig. 1(b), the $Q$ values achieved by the algorithms match their accuracies: *WS* has the highest modularity, followed by *K-5*, *K-4*, ..., and the Newman algorithm at last. A third measure, the number of times an algorithm predicted $k$ correctly, also shows that *WS* > *K-5* > $\cdots$ > *K-2* > *NM* (data not shown). The *CNM* algorithm has an accuracy similar to *K-2* for smaller $p_{out}$, but its accuracy drops significantly when $p_{out}$ increases.

## 4.2 Real-world networks

We further tested our method on several real-world networks. These include an acquaintance network in a Karate club [35], the opponent network of American NCAA Division I college football teams in the year 2000 [12], a co-performing network of Jazz Bands [13], a protein-protein interaction network of *E. coli* [25], the Autonomous Systems topology of the Internet [10], and a collaboration net-

**Table 1. $Q$ values for real-world networks.**

| Network | $n$ | $m$ | $K^*$ | $K_{max}$ | $Q$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | *WS* | *K-2* | *K-3* | *K-4* | *K-5* | *NM* | *CNM* | Best |
| Karate | 34 | 78 | 4 | 8 | 0.420 | 0.390 | 0.420 | 0.420 | 0.420 | 0.393 | 0.383 | 0.420 [33] |
| Football | 115 | 613 | 10 | 20 | 0.602 | 0.524 | 0.600 | 0.596 | 0.590 | 0.493 | 0.577 | |
| Jazz | 198 | 5484 | 4 | 8 | 0.439 | 0.444 | 0.444 | 0.439 | 0.439 | 0.394 | 0.439 | 0.445 [7] |
| PPI | 1440 | 6223 | 133 | 200 | 0.362 | 0.332 | 0.344 | 0.348 | 0.364 | 0.341 | 0.337 | |
| Internet | 3015 | 5156 | 52 | 100 | 0.604 | 0.594 | 0.600 | 0.601 | 0.601 | 0.524 | 0.620 | |
| Physicists | 27519 | 60793 | - | 600 | - | 0.734 | 0.738 | 0.739 | 0.743 | - | 0.659 | 0.723 [21] |

$K_{max}$: maximal number of communities for *WS*. $K^*$: number of communities returned by *WS*. The last column are the best $Q$ values achieved by existing methods in the literature, and references to the methods.

**Table 2. Total CPU time (seconds).**

| Network | *WS* | *K-2* | *K-3* | *K-4* | *K-5* | *NM* | *CNM** |
|---|---|---|---|---|---|---|---|
| Karate | 0.3 | 0.3 | 0.3 | 0.3 | 0.4 | 0.1 | 0.02 |
| Football | 1.1 | 0.7 | 0.6 | 0.7 | 1.1 | 0.3 | 0.04 |
| Jazz | 0.5 | 0.6 | 0.7 | 0.7 | 0.9 | 0.3 | 0.06 |
| PPI | 8k | 40 | 26 | 31 | 23 | 58 | 0.8 |
| Internet | 3k | 37 | 27 | 22 | 23 | 172 | 63 |
| Physicists | - | 6k | 3k | 2k | 2k | - | 283 |

*A significant difference between *CNM* and the other algorithms here is that *CNM* was implemented in C, while all the other algorithms compared here were implemented in MATLAB m-files.

work of physicists [19]. As shown in Table 1, the *WS* algorithm usually returns community structures with the highest $Q$ value. Although *Kcut* with $l = 2$ often performed poorly, *Kcut* with $l \geq 3$ can usually achieve $Q$ values as good as that by *WS*, whereas with a much reduced running time. Moreover, for the three networks (Karate, Jazz, Physicists) that have been analyzed by others, *Kcut* can find modularity values that are comparable to or better than the best known ones. The *NM* algorithm (without the refining step) and the *CNM* algorithm usually have much worse accuracy comparing to *WS* and *Kcut*. The *WS* and Newman algorithms failed to finish on the physicist network, due to their excessive running time or memory usage.

In addition, the communities returned by *Kcut* are often very close to the known communities if they are available. For example, for the Karate club network, *Kcut* precisely predicted the actual separation of the club caused by a dispute among its members [12]. For the football network, *Kcut* correctly revealed the official NCAA conference structure of the football teams [12], except for a few teams that do not belong to any conference. Because of space limit, we omit the detailed results here, and focus on networks of real applications, which will be shown in Section 5.

### 4.3 Running time

Table 2 shows the running time of the four algorithms on the six real-world networks. Table 3 shows the time spent on *eigs* and $k$-means by *WS*, *Kcut* and *NM*. *CNM* is based on a different rationale and does not have these two components. As shown in Table 2, although *WS* is efficient for small networks of up to a few hundred of vertices, it is very inefficient on large networks. The *Kcut* algorithm, on the other hand, can handle networks of several thousand of vertices in less than half minute. It appears in Table 2 that *CNM* is the most efficient, especially for small networks. At least part of the reason is that *CNM* was implemented in the C language, while the other three algorithms were all implemented in MATLAB M-files. M-files are interpreted at run time, and therefore have higher overhead.

Also observe that *Kcut* is often faster with $l = 3, 4, 5$ than with $l = 2$. Based on the analysis in Section 3.1, the time *Kcut* spent in *eigs* is approximately linear to $l/\ln l$, which reaches its minimum at $l = 3$. In contrast, the time *Kcut* spent on $k$-means is proportional to $l^2/\ln l$, which is monotonically increasing for $l \geq 2$. The experimental results in Table 3 partially support the theoretical analysis. For large networks, the total running time of *Kcut* is dominated by *eigs*. Therefore, *Kcut* can take advantage of a slightly increased $l$ to reduce its running time. When $l$ becomes too large, however, the running time of both components increases, and the efficiency of *Kcut* may degrade.

## 5 Real applications

Community discovery algorithms have a broad range of applications in many disciplines. Here, we demonstrate the utility of our algorithm on three real applications in biology, medicine, and social science, respectively.

### 5.1 Communities in a gene network

It is believed that biological systems are modular, consisting of groups of bio-molecules responsible for different functions, such as metabolism, reproduction, and signal transduction, etc. The relationships between bio-molecules have been mapped to several types of networks, including gene regulatory networks [18] and protein interaction networks [25]. Therefore, it is interesting to test whether these networks contain communities, and whether the community structure is related to the functional classification of

**Table 3. CPU time (seconds) for program components.**

| Network | *WS* | | *K-2* | | *K-3* | | *K-4* | | *K-5* | | *NM* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Karate | 0.08 | 0.16 | 0.16 | 0.11 | 0.1 | 0.11 | 0.1 | 0.2 | 0.08 | 0.22 | 0.11 |
| Football | 0.1 | 0.81 | 0.33 | 0.22 | 0.21 | 0.23 | 0.29 | 0.36 | 0.23 | 0.7 | 0.28 |
| Jazz | 0.11 | 0.29 | 0.26 | 0.06 | 0.31 | 0.23 | 0.25 | 0.3 | 0.25 | 0.5 | 0.25 |
| PPI | 14 | 7857 | 34 | 3 | 20 | 4 | 18 | 7 | 11 | 8 | 53 |
| Internet | 12 | 2892 | 31 | 3 | 19 | 5 | 14 | 6 | 11 | 9 | 150 |
| Physicists | - | - | 5353 | 79 | 2451 | 109 | 1473 | 152 | 1473 | 170 | - |

For *WS* and *Kcut*, the first and second columns are the time taken by *eigs* and $k$-means, respectively. The last column is the time spent on *eigs* in the Newman algorithm.

**Table 4. Enriched GO functional terms in communities.**

| Community | size | Most significant GO term | $f$ | $F$ | $p$ | $p^{HCL}$ |
|---|---|---|---|---|---|---|
| 1 | 460 | ribosome biogenesis | 32.8% | 2.9% | 1.90E-138 | 4.30E-120 |
| 2 | 159 | nuclear mRNA splicing, via spliceosome | 46.5% | 1.4% | 1.10E-105 | 4.20E-89 |
| 3 | 272 | protein biosynthesis | 62.9% | 11.1% | 9.80E-97 | 2.10E-65 |
| 4 | 334 | transcription | 40.4% | 6.8% | 8.80E-72 | 2.70E-17 |
| 5 | 172 | response to DNA damage stimulus | 41.9% | 2.6% | 9.30E-71 | 1.80E-31 |
| 6 | 449 | secretory pathway | 21.8% | 2.9% | 2.20E-62 | 1.00E-50 |
| 7 | 226 | nucleocytoplasmic transport | 23.9% | 1.6% | 5.60E-50 | 9.90E-34 |
| 8 | 346 | cell cycle | 29.5% | 5.6% | 4.80E-46 | 1.70E-57 |
| 9 | 118 | nitrogen compound metabolism | 39.8% | 3.3% | 2.70E-37 | 3.90E-18 |
| 10 | 172 | proteolysis during cellular protein catabolism | 23.3% | 1.8% | 3.90E-32 | 6.30E-20 |
| 11 | 90 | oxidative phosphorylation | 24.4% | 0.6% | 1.30E-28 | 5.00E-05 |
| 12 | 94 | RNA 3'-end processing | 17.0% | 0.3% | 1.50E-23 | 1.10E-09 |
| 13 | 9 | pre-replicative complex formation and maintenance | 77.8% | 0.2% | 1.70E-17 | 6.50E-08 |
| 14 | 14 | protein amino acid N-linked glycosylation | 64.3% | 0.6% | 1.30E-15 | 5.50E-14 |
| 15 | 205 | protein targeting to mitochondrion | 9.3% | 0.7% | 1.10E-14 | 4.20E-09 |
| 16 | 585 | carbohydrate metabolism | 10.1% | 3.1% | 3.90E-14 | 9.10E-06 |
| 17 | 8 | histone methylation | 75.0% | 0.2% | 1.40E-13 | 4.80E-05 |
| 18 | 18 | mRNA catabolism | 50.0% | 0.8% | 1.60E-13 | 8.30E-07 |
| 19 | 42 | biological process unknown | 73.8% | 22.0% | 9.40E-11 | 3.70E-09 |
| 20 | 6 | chromatin assembly | 100.0% | 1.2% | 1.30E-10 | 6.20E-12 |

$f$, $F$: percentage of genes in the community or genome associated with the GO term; $p$: $p$-value for the enrichment of a GO term in a community; $p^{HCL}$: $p$-value for the enrichment of the same GO term in a gene clusters identified by hierarchical clustering [17].

the molecules.

We applied the *Kcut* algorithm to study a network of ≈4500 genes in the yeast *S. cerevisiae*, obtained by integrating a number of heterogeneous biological data sources [17]. The vertices in the network are genes, and an edge between two genes indicates that they may be involved in some common biological process supported by literature or experimental evidence such as gene expression microarray or protein interaction data [17]. The network may contain errors, and the edges may have different meaning, depending on the data source from which they were derived.

Applying *Kcut*, we obtained 47 communities, whose sizes range from 2 to 585. Our analysis below is focused on the 36 communities that contain more than five genes. To understand the functions of these communities, we checked the enrichment of Gene Ontology (GO) terms [31], which are a set of controlled vocabulary used to annotate genes functions, for the genes within each community. Enrichment is computed using hyper-geometric test [3]. Briefly, suppose that a community contains $m$ genes, $n$ of which are annotated by a function *T*, and that $N$ out of $M$ genes in the whole genome are annotated by *T*. the hyper-geometric test measures the probability that we would expect at least $n$ genes annotated by *T* if we had randomly drawn $m$ genes from the genome. A low probability indicates that the community is significantly enriched with *T*. To account for multiple testing, we corrected the $p$-values using a Bonferroni procedure [3], and used $p$-value = 0.05 as a threshold for the significance test.

All except one of the 36 communities contain significantly enriched GO terms. Overall, these communities contain 2123 enriched GO terms. In contrast, using a hierarchical clustering method, Lee et. al. obtained 55 clusters [17]. Among them, 32 clusters contributed to a total of 1746 significantly enriched GO terms, much fewer than that in *Kcut*. Figure 2 compares the $p$-values for the GO terms that are enriched in both *Kcut* and hierarchical clustering. As shown, most GO terms are more significantly enriched in *Kcut* than in hierarchical clustering, indicating the advantage of our methods against conventional graph clustering methods.

Table 4 lists 20 communities that have the most significantly enriched GO terms. Many communities contain very significantly enriched GO terms. For example, about one third of the genes in community 1 have functions in ribosome biogenesis, an 11-fold enrichment that are unlikely due to chance ($p <$ 2e-138). About 40% of genes in com-

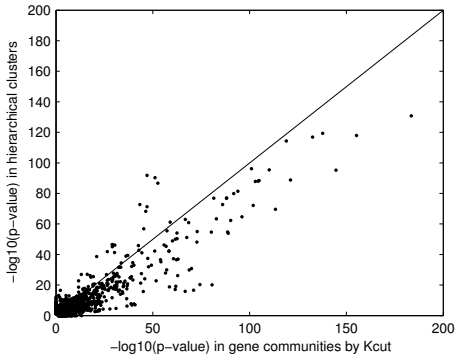


**Figure 2. Comparison of GO enrichment**

munity 4 are related to transcription ($p <$ 9e-72). In contrast, only 21% of genes in the corresponding hierarchical cluster are annotated by this term ($p <$ 3e-17).

Since genes in the same community often share functions, the community structure may be used to predict gene functions. For example, community 19 contains 42 genes. The most significant function category is "biological process unknown" (group A, 31 genes, $p <$ 1e-10). The second most significant GO term is "telomerase-independent telomere maintenance" (group B, 5 genes, $p <$ 1e-6). To test whether the 31 unknown genes are also related to telomere maintenance, we searched the Saccharomyces Genome Database (`http://www.yeastgenome.org/`) and other experimental data for more information about these genes. Using whole-genome ChIP-chip data [18], we found that all 5 group B genes are regulated by at least three of the following four transcription factors: YAP5, GAT3, MSN4, and PDR1. Interestingly, many of the 31 group B genes, but none of the other genes, are also regulated by the four transcription factors: 18 by YAP5 ($p <$1e-13), 16 by GAT3 ($p <$6e-14), 8 by MSN4 ($p <$2e-8) and 9 by PDR1 ($p <$2e-8). In addition, all the 5 group A genes and 10 of the group B genes have helicase activity ($p <$ 2e-17). Furthermore, the majority of the 31 unknown genes are located in the telomere regions of chromosomes. These seem to be strong evidences that these unknown genes are very likely related to telomere maintenance functions.

## 5.2 Automatic detection of tumor types

An accurate classification of tumor cells is crucial for effective therapy [9]. Traditionally, tumors have been classified by their morphologic appearance, which is, unfortunately, often very subjective. Furthermore, tumors with similar histological features often respond very differently to chemotherapy [2]. To get around this problem, a promising alternative or complementary strategy is to classify tumors based on their genetic profiles, i.e. the activity of hundreds or thousands of genes that are involved in the disease. Most of the existing tumor classification approaches are based on supervised learning, such as support vector machines or decision trees, which aimed at identifying genetic features to distinguish two or more known tumor types [16, 27].

Here, we ask whether it is possible to automatically classify tumor samples into distinct types or subtypes, without knowing the sample labels or even the number of labels. This unsupervised learning approach has a few advantages over supervised learning methods. First, the existing tumor classifications are based on histological features, which may be unreliable themselves. Second, using unsupervised learning, we may be able to discover novel tumor sub-types that have not been characterized by histological features previous. On the other hand, it is crucial to confirm whether the automatically discovered tumor types are biologically meaningful and can provide useful information for understanding the disease or for improving its treatment.

In this study, we choose to focus on lymphoma, a family of tumors involving cells of the immune system, because of the large number of people it affects and the relative abundance of microarray data studying the diseases. We obtained the data from [2], which contains the expression data of 4026 genes for 96 samples belonging to nine cell types, including three different types of tumors, i.e. diffuse large B cell lymphoma (DLBCB), chronic lymphocytic leukemia (CLL), and follicular lymphoma (FL), as well as normal B and T cells at different stages of cell differentiation [2].

From the gene expression data, we constructed a nearest-neighbor network of samples. Briefly, we first computed a pair-wise Pearson correlation coefficient between every two samples based on their gene expression patterns. We then connect each sample to five other samples whose correlation to the sample is the highest. This resulted in a network where each sample is connected to five or more samples. The edges are weighted by the correlation coefficients. Edges with a weight below a cutoff 0.2 are removed to avoid false connections.

Applying the *Kcut* algorithm to the network, we identified eight communities, each of which contains at most two cell types (Table 5). Furthermore, the results are almost invariant when we changed the number of nearest neighbors for each sample between 3 and 7, indicating very stable community structures among the samples. As shown in Table 5, FL cells and activated blood B cells are perfectly classified into their own communities. Community 6 contains all the blood T cells and only one DLBCB cell. CLL and resting B cells are clustered into a single community, which is not surprising since the former has a very low proliferation rate, similar to the latter [2]. Community 1 contains almost half transformed cell lines and half DLBCB samples. With a closer inspection, however, we found that three of the DLBCB samples in this community

**Table 5. Communities of cell samples**

| Community | Samples | Cell type |
|---|---|---|
| 1 | OCI-Ly3*, OCI-Ly10*, DLCL-0042*, OCI-Ly1*, WSU1, Jurkat, U937, OCI-Ly12, OCI-Ly13.2, SUDHL5, DLCL-0041* | Transformed cell lines<br>*: DLBCB |
| 2 | DLCL-0030, DLCL-0004, DLCL-0029, Tonsil GC B*, Tonsil GC Centroblasts*, SUDHL6, DLCL-0008, DLCL-0052, DLCL-0034, DLCL-0051, DLCL-0032, DLCL-0018, DLCL-0037, DLCL-0020, DLCL-0003, DLCL-0033 | DLBCB<br>*:, GCB |
| 3 | DLCL-0011, DLCL-0006, DLCL-0049, Tonsil*, DLCL-0039, Lymph Node*, DLCL-0001, DLCL-0015, DLCL-0026, DLCL-0005, DLCL-0023, DLCL-0027, DLCL-0024, DLCL-0013, DLCL-0002, DLCL-0016, DLCL-0014, DLCL-0048 | DLBCB<br>*:, Lymph node / tonsil |
| 4 | DLCL-0007, DLCL-0031, DLCL-0036, DLCL-0010, DLCL-0025, DLCL-0040, DLCL-0017, DLCL-0028, DLCL-0012, DLCL-0021 | DLBCB |
| 5 | Blood T:Adult CD4+ Unstim, Blood T:Adult CD4+ I+P Stim, Cord Blood T:Neonatal I+P Stim, Blood T:Neonatal CD4+ Unstim, Thymic T:Fetal CD4+ Unstim, Thymic T:Fetal CD4+ I+P Stim, DLCL-0009* | Blood T cells<br>*:, DLBCB |
| 6 | Blood B:memory*, Blood B:naive*, Blood B*, Cord Blood B*, CLL-60, CLL-68, CLL-9, CLL-14, CLL-51, CLL-65, CLL-71;Richter's, CLL-71, CLL-13, CLL-39, CLL-52 | CLL<br>*: Resting blood B. |
| 7 | Blood B:anti-IgM+CD40L low 48h, Blood B:anti-IgM+CD40L high 48h, Blood B:anti-IgM+CD40L 24h, Blood B:anti-IgM 24h, Blood B:anti-IgM+IL-4 24h, Blood B:anti-IgM+CD40L+IL-4 24h, Blood B:anti-IgM+IL-4 6h, Blood B:anti-IgM 6h, Blood B:anti-IgM+CD40L 6h, Blood B:anti-IgM+CD40L+IL-4 6h | Activated blood B cells |
| 8 | FL-9, FL-9;CD19+, FL-12;CD19+, FL-10;CD19+, FL-10, FL-11, FL-11;CD19+, FL-6;CD19+, FL-5;CD19+ | FL |

(OCI-Ly1, Ly3, and Ly10) are actually laboratory cultivated cell lines rather than samples from real patients, which may be the reason that they are grouped together with the transferred cell lines. The GCB and Lymph node/tonsil cells are grouped together with DLBCB as expected, but a discussion is out of the scope of this paper [2]. Therefore, our algorithm has detected most of the cell types successfully, without any knowledge about the samples.

Furthermore, the majority of the DLBCB samples are grouped into three communities (2, 3, and 4). It is known that not all DLBCB tumors are equal: 40% of patients respond well to chemotherapy and have prolonged survival, while the others have a much shorter survival time after the treatment [2]. To test whether the three communities correspond to different tumor subtypes, we counted the rate of survival after chemotherapy for the patients within each community. Remarkably, the patients in community 2 have a survival rate much higher than average. The median survival length for the patients in the three communities are 71.3, 23, and 12.5 months, respectively. In fact, 9/11 (82%) patients in community 2 lived more than five and half years after the treatment, while only 5/26 (19%) patients in communities 3 and 4 lived this long.

Furthermore, we combined our classification of DLBCB samples with the International Prognostic Index (IPI), which is a clinical tool developed by oncologists to predict the prognosis of lymphoma patients [2]. In general, the patients with IPI $> 2$ have a much lower survival rate (21.4%) than those with IPI $\leq 2$ (62.5%). By combining our classifications, we find that all (8/8) the community-2 patients having IPI $\leq 2$ survived more than 5 years after the therapy. In contrast, for the patients in communities 3 and 4 with IPI $\leq 2$, only 23%(3/13) survived more than 5 years. Furthermore, only one of the eight patients in communities 3 and 4 with IPI $> 2$ survived more than 5 years. These results indicate that by combining genetic and clinical information, we may be able to better predict the outcome following the chemotherapies.

### 5.3 Communities in newsgroups

As the last application, we analyzed the community structure in the Usenet, the network of topic-oriented newsgroups on the Internet. Most existing attempts in analyzing the structure of Usenet focused on the semantic properties of newsgroups, for example, by clustering or classifying the messages according to their contents [24, 28]. Here, we are interested in studying the social network among newsgroup users. We ask whether it is possible to identify groups of users with common interests without looking at the actual content of their messages. To this end, we use what we call a message-replying network, where each vertex is a unique user, and an edge between two users indicates that one has replied a message of the other. The links are weighted by the number of times the communication occurred between them. The network is made into bidirectional, i.e. we ignore the directions of edges.

We downloaded from UCI KDD Archive website (`http://kdd.ics.uci.edu/`) a data set containing 20,000 messages from 20 newsgroups. We extracted the email address in the "From:" field of each message, and the first email address in the text in a sentence like "xxx@yyy writes:", which indicates that the current message is a response to an earlier message of xxx@yyy. From these messages, we identified a total of 9746 unique users, 5468 of which form a giant connected component. Our experiment is focused on this giant component, which contains 12306

edges. The degrees of the vertices follow a power-law distribution with $r = 3.6$.

Applying *Kcut*, we obtained 48 communities with a $Q$ value = 0.9116, indicating a very strong community structure. The sizes of the communities are highly diverse, ranging from 9 to 371. It is important to note that the network structure is the only information used by our algorithm for community detection. Semantic information such as the content of a message, as well as to which newsgroup a message was posted, are only used to analyze the communities.

To check whether the members of each community indeed have similar interests, we first checked the posting history of the users. Figure 3 shows the messages posted in the 20 newsgroups by the users in the 25 largest communities, where each row represents a user, and each column represents a newsgroup. The horizontal line shows the boundary of communities. The data is normalized so that each row has a unit length. As shown, the users in each community post messages primarily to a particular newsgroup, meaning that the detected communities indeed correspond to common interests or topics. To have a rough estimation of the performance, we assume that each newsgroup is a proxy of a community, and each user belongs to a newsgroup/community where he/she has posted the most messages. This is of course a very crude estimation, since many users post to multiple newsgroups equally. It is also probably not true that the number of communities is equivalent to the number of newsgroups, since the users within a single newsgroup may have different interests, while some newsgroups often cross-link messages. Nevertheless, 78% of the users can be assigned to a correct community by looking at the message-replying network alone, which is not perfect, but very encouraging.

To test whether the users in different communities have different interests, we computed statistics of the words used by the members of each community. We first pooled together all messages in the 20 newsgroups, and counted the background frequency that each word is used. We then combined the messages posted by the members of a community, and computed the target frequency of each word in the community. We defined a word as a significant keyword for a community if its target frequency is ten times higher than its background frequency. We then sorted the keywords in each community according to their absolute target frequency. The top ten most frequently used keywords for each community are listed in Table 6.

As can be seen, most communities have their unique combination of keywords highly related to the newsgroups that they frequently send messages to. For example, the users in community 1 frequently post to `rec.motorcycles`, and the most significant keywords in the community are "bike" and "ride". Sometimes users in different communities may post to the same newsgroups. For

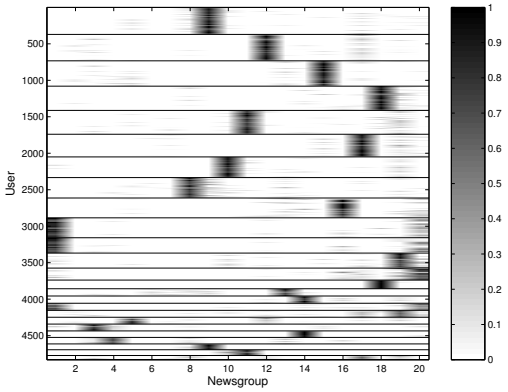


**Figure 3. User posting history.** The 20 newsgroups, from left to right, are alt.atheism, comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x, misc.forsale, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, soc.religion.christian, talk.politics.guns, talk.politics.mideast, talk.politics.misc, and talk.religion.misc.

example, users in communities 10, 11 and 17 all post to `alt.atheism`, and users in communities 4 and 14 often post to `talk.politics.mideast`. It can be seen that, however, these communities are represented by very different keywords, indicating the existence of sub-structures within these newsgroups (Table 6).

Furthermore, we also observed that users in some communities frequently post to two or more newsgroups, which indicates some relationship between these newsgroups (Fig 3). Examples include newsgroups 1 (`alt.atheism`) and 20 (`talk.religion.misc`), 3 (`comp.os.ms-windows.misc`) and 4 (`comp.sys.ibm.pc.hardware`), 17 (*talk.politics.guns*) and 19 (`talk.politics.misc`). The relationships between these newsgroups are obvious from the names of the newsgroups.

## 6 Conclusions

We have developed a fast algorithm, *Kcut*, for identifying community structures in large networks. Our approach is based on a greedy optimization of a modularity function $Q$. Unlike previous methods, *Kcut* is not restricted to bi-partitions, but considers all $k$-way partitions for a small range of $k$. We have found that this relaxation not only improves the quality of the identified communities, but also increases the efficiency of the algorithm. We have demonstrated the performance of our method on a variety of random and real-world networks. Compared to the existing approaches, *Kcut* can find better $Q$ values than other greedy approaches, and has an accuracy comparable to that of a much slower exhaustive search method.

In addition, we have applied our method to several real problems in three different fields: biology, medicine and social science. For all cases, our algorithm is able to detect significant and meaningful community structures, and

**Table 6. Keywords within each community**

| Community | Size | Newsgroup | Keywords |
|---|---|---|---|
| 1 | 371 | rec.motorcycles | bike,ride,helmet,bikes,riding,dog,motorcycle,viking,passenger,mph |
| 2 | 362 | sci.crypt | clipper,encryption,keys,escrow,algorithm,secure,encrypted,phones,proposal,wiretap |
| 3 | 347 | sci.space | launch,orbit,mission,shuttle,pat,solar,spacecraft,moon,flight,sky |
| 4 | 332 | talk.politics.mideast | holocaust,museum,attacks,civilians,memorial,bullock,occupied,territories,territory |
| 5 | 327 | rec.sport.hockey | team,hockey,season,insurance,player,blues,teams,cup,ice,playoffs |
| 6 | 311 | talk.politics.guns | gun,guns,weapons,firearms,deaths,handgun,homicides,concealed,boulder,handguns |
| 7 | 283 | rec.sport.baseball | convention,political,hit,players,party,baseball,average,clutch,sept,parties |
| 8 | 280 | rec.autos | car,cars,engine,dealer,mustang,ford,flat,rear,bird,mph |
| 9 | 272 | soc.religion.christian | sin,homosexual,scripture,marriage,worship,baptism,pope,spiritual,sabbath,sinful |
| 10 | 272 | alt.atheism | moral,objective,morality,perry,served,immoral,leftover,truelove,prophecy,messiah |
| 11 | 211 | alt.atheism | atheism,atheists,atheist,hatching,hens,fallacy,theism,theists,dollar,morals |
| 12 | 205 | talk.politics.misc | sex,sexual,homosexual,gay,homosexuals,male,partners,associate,gravity |
| 13 | 165 | talk.religion.misc | magi,commandments,conscious,hanging,teaches,persecution,guilt,contradictions |
| 14 | 119 | talk.politics.mideast | turks,soviet,genocide,turkey,extermination,population,ottoman,mountain,roads |
| 15 | 99 | sci.electronics | wire,wiring,ground,circuit,neutral,cable,outlets,battery,hot,ham,wires,connected |
| 16 | 99 | sci.med | vitamin,universe,physical,patients,yeast,doctor,diet,medicine,treatment,clinical |
| 17 | 97 | alt.atheism | evolution,theory,gravity,spirit,holy,theories,creed,jack,fox,grace,insert,rich |
| 18 | 95 | talk.politics.misc | warrant,drugs,murder,knock,barrel,officers,raid,govt,reno,founding |
| 19 | 94 | comp.sys.mac.hardware | apple,keyboard,option,energy,duo,coprocessor,macs,movie,militia,playback |
| 20 | 93 | comp.os.ms-windows.misc | windows,win,modem,zip,swap,apps,fonts,ports,icon,mickey |
| 21 | 90 | sci.med | cause,diet,corn,patients,thyroid,blood,doctor,treatment,eye,disease |
| 22 | 89 | comp.sys.ibm.pc.hardware | bit,bus,mac,fast,memory,controller,drives,mode,interface,faster |
| 23 | 79 | rec.motorcycles | bike,riding,dog,rider,technique,helmet,motorcycle,evil,difficult,tom |
| 24 | 79 | rec.sport.hockey | game,team,hockey,players,league,wings,baseball,leafs,player,teams |
| 25 | 58 | talk.politics.guns | gun,firearms,firearm,section,crime,guns,weapon,license,military,committee |

the community structures can provide important information about the systems of interest, which may have many practical applications.

## Acknowledgments

This research was supported in part by NSF grants ITR/EIA-0113618 and IIS-0535257 and a grant from Monsanto Company to W.Z.

## References

[1] R. Albert and A. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.

[2] A. Alizadeh, M. Eisen, R. Davis, C. Ma, I. Lossos, A. Rosenwald, J. Boldrick, H. Sabet, T. Tran, X. Yu, J. Powell, L. Yang, G. Marti, T. Moore, J. H. J, L. Lu, D. Lewis, R. Tibshirani, G. Sherlock, W. Chan, T. Greiner, D. Weisenburger, J. Armitage, R. Warnke, R. Levy, W. Wilson, M. Grever, J. Byrd, D. Botstein, P. Brown, and L. Staudt. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–11, 2000.

[3] D. Altman. *Practical Statistics for Medical Research.* Chapman & Hall/CRC, 1991.

[4] J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. Wallace. Discovering hidden groups in communication networks. 2nd NSF/NIJ Symposium on Intelligence and Security Informatics., 2004.

[5] A. Clauset and et. al. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004.

[6] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas. Comparing community structure identification. *J. Stat. Mech.*, page P09008, 2005.

[7] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72:027104, 2005.

[8] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, pages 147–153, 2003.

[9] D. Ellis, M. Eaton, R. Fox, S. Juneja, A. Leong, J. Miliauskas, D. Norris, D. Spagnolo, and J. Turner. Diagnostic pathology of lymphoproliferative disorders. *Pathology*, 37:434–56, 2005.

[10] M. Faloutsos and et. al. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[11] P. Fjallstrom. Algorithms for graph partitioning: A survey. Linkoping Electron. Atricles in Comput. and Inform. Sci., 1998.

[12] M. Girvan and M. Newman. Community structure in social and biological networks. *Proc Natl Acad Sci U S A*, 99:7821–6, 2002.

[13] P. Gleiser and L. Danon. Community structure in jazz. *Advances in Complex Systems*, 6:565, 2003.

[14] R. Guimera and L. N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.

[15] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Tracking evolving communities in large linked networks. *Proc Natl Acad Sci U S A*, 101 Suppl 1:5249–53, 2004.

[16] J. Jaeger, D. Koczan, H. Thiesen, S. Ibrahim, G. Gross, R. Spang, and M. Kunz. Gene expression signatures for tumor progression, tumor subtype, and tumor thickness in laser-microdissected melanoma tissues. *Clin Cancer Res*, 13:806–15, 2007.

[17] I. Lee, S. Date, A. Adai, and E. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306:1555–8, 2004.

[18] T. Lee, N. Rinaldi, F. Robert, D. Odom, Z. Bar-Joseph, G. Gerber, N. Hannett, C. Harbison, C. Thompson, I. Simon, J. Zeitlinger, E. Jennings, H. Murray, D. Gordon, B. Ren, J. Wyrick, J. Tagne, T. Volkert, E. Fraenkel, D. Gifford, and R. Young. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298:799–804, 2002.

[19] M. Newman. The structure of scientific collaboration networks. *Proc Natl Acad Sci USA*, 98:404–409, 2001.

[20] M. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[21] M. Newman. Modularity and community structure in networks. *Proc Natl Acad Sci USA*, 103:8577–82, 2006.

[22] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys Rev E*, 69:026113, 2004.

[23] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[24] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.

[25] L. Salwinski, C. Miller, A. Smith, F. Pettit, J. Bowie, and D. Eisenberg. The database of interacting proteins: 2004 update. *Nucleic Acids Res*, 32:D449–51, 2004.

[26] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:888–905, 2000.

[27] M. Shipp, K. Ross, P. Tamayo, A. Weng, J. Kutok, R. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. Pinkus, T. Ray, M. Koval, K. Last, A. Norton, T. Lister, J. Mesirov, D. Neuberg, E. Lander, J. Aster, and T. Golub. Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nat Med*, 8:68–74, 2002.

[28] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proc. of ACM SIGIR conf on Research and development in information retrieval*, pages 208–215, New York, NY, USA, 2000. ACM Press.

[29] V. Spirin and L. Mirny. Protein complexes and functional modules in molecular networks. *Proc Natl Acad Sci U S A*, 100:12123–8, 2003.

[30] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 2005.

[31] The Gene Ontology Consortium. The gene ontology (go) database and informatics resource. *Nucleic Acids Res*, 32, 2004.

[32] D. Verma and M. Meila. A comparison of spectral clustering algorithms. Technical report, Univ. of Washington, 2003.

[33] S. White and P. Smyth. A spectral clustering approach to finding communities in graph. In *SIAM Data Mining*, 2005.

[34] D. Wilkinson and B. Huberman. A method for finding communities of related genes. *Proc Natl Acad Sci U S A*, 101 Suppl 1:5241–8, 2004.

[35] W. Zachary. An information flow model of conflict and fission in small groups. *J. Anthropol. Res.*, 33:452–473, 1993.