On Structural Analysis of Large Networks

Nurcan Yuruk
University of Arkansas at Little Rock
nxyuruk@ualr.edu

Xiaowei Xu
University of Arkansas at Little Rock
1xiaowei.xu@gmail.com

Thomas A. J. Schweiger
Acxiom Corporation
Tom.Schweiger@acxiom.com

ABSTRACT

Containing much valuable information, networks such as the World Wide Web, social networks and metabolic networks draw increasingly attention in scientific communities. Network clustering (or graph partitioning) is the discovery of underlying clusters of related vertices in networks. But beyond organizing vertices into clusters of peers is the question of what role each vertex play in the network. This paper presents some new ways of uncovering underlying structures, including the roles that vertices play in the network. Identifying vertex roles is useful for applications such as viral marketing and epidemiology. For example, hubs are responsible for spreading ideas or disease. We applied our algorithm to analyze some real networks. The results demonstrate a superior performance over other methods such as modularity-based algorithms.

1. INTRODUCTION

Networks are ubiquitous. Common networks include social networks, the World Wide Web, and computer networks. A network consists of a set of vertices interconnected by edges. A vertex represents some real entities such as a person, website, or piece of networking hardware. An edge connects two vertices if they have some relationship such as a friendship, hypertext link, or wired connection. In such networks, each vertex plays a role. Some vertices are members of clusters; a group of peers in a social network or a group of related websites in the WWW are examples. Some vertices are hubs that bridge many clusters but don't belong strongly to any one cluster; for example politicians tend to play such a role in social networks, and websites like wikipedia.org are clearing-houses for all kinds of information. Some vertices represent outsiders that have only weak associations with any cluster; for example a loner in a social network, or a parked domain on the internet.

To illustrate these points further, consider the network in Figure 1. If might confidently consider the vertices {0,1,2,3,4,5} and {7,8,9,10,11,12} to be clusters of peers. Vertex 6 is difficult to classify. It could arguably belong to either cluster or to none. It is an example of a hub. Likewise, vertex 13 is weakly connected to a cluster. It is an example of an outsider.

Network clustering (or graph partitioning) is the detection of structures like those in Figure 1, and it is drawing increased attention and application in computer science [1][2], physics [8], and bioinformatics [3]. Various methods have been developed. They tend to partition on the principle that clusters should be sparsely connected, but the vertices within each cluster should be densely connected. Modularity-based algorithms [3][8][9] and normalized cut [1][2] are successful examples. However, they do not distinguish the roles of vertices.

The modularity-based algorithm [9] will cluster the network in Figure 1 into two clusters: one consisting of vertices 0 to 6 and the other consisting of vertices 7 to 13. It doesn't isolate vertex 6 or vertex 13.

The identification of hubs gives valuable information. For example, hubs in the WWW are deemed authoritative information sources among web pages [4], and hubs in social networks are believed to play a crucial role in viral marketing [5] and epidemiology [6].

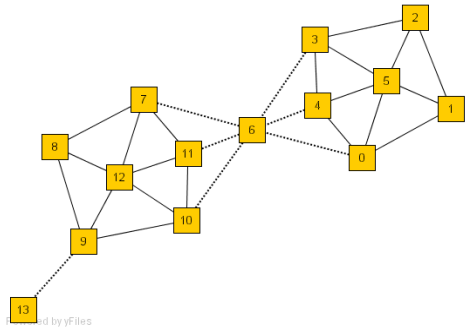


Figure 1. A Network with 2 Clusters, a Hub and an Outsider.

In this paper, we propose a new method for network clustering. The goal of our method is to find clusters, hubs, and outsiders in large networks. To achieve this goal, we use the neighborhood of the vertices as clustering criteria instead of only their direct connections. Vertices are clustered by how they share neighbors. Doing so makes sense when you consider the detection of communities in large social networks. Two people who share many friends should be clustered in the same community.

Referring to figure 1, consider vertices 0 and 5, which are connected by an edge. Their neighborhoods are the vertex sets {0, 1, 4, 5, 6} and {0, 1, 2, 3, 4, 5}, respectively. They share many neighbors and thus are reasonably clustered together. In contrast, consider the neighborhoods of vertex 13 and vertex 9. These two vertices are connected, but share only few common neighbors, i.e. {9, 13}. Therefore, it is doubtful that they should be grouped together. The situation for vertex 6 is a little different. It has many neighbors, but they are sparsely interconnected.

Our algorithm identifies two clusters, {0, 1, 2, 3, 4, 5} and {7, 8, 9, 10, 11, 12}, and isolates vertex 13 as an outsider and vertex 6 as a hub.

Our algorithm has the following features:

- It detects clusters, hubs, and outsiders by using the structure and the connectivity of the vertices as clustering criteria.
- It is efficient. It clusters the given network by visiting each vertex exactly once.

Through theoretical analysis and experimental evaluation we will demonstrate that our algorithm finds meaningful clusters and identifies hubs and outsiders in very large networks,

With respect of efficiency, our algorithm's running time on a network with n vertices and m edges is $O(m)$. In contrast, the

1530-1605/08 $25.00 © 2008 IEEE

running time of the fast modularity-based algorithm [9] is $O(md \log n)$.

The paper is organized as follows. We formulize the notion of structure-connected cluster (SCC) in section 2. We devise an algorithm to find SCC in section 3. We give a computation complexity analysis of our algorithm in section 4. We compare our algorithm to the fast modularity-based algorithm in section 5. We review the related work in section 6. Finally, we present our conclusions and suggest future work in section 7.

2. THE NOTION OF STRUCTURE-CONNECTED CLUSTER

Our goal is both to cluster network optimally and to identify and isolate hubs and outsiders. Therefore, both connectivity and local structure is used in our definition of optimal clustering. In this section, we formulize the notion of a structure-connected cluster, which extends that of a density-based cluster [7] and can distinguish clusters, hubs, and outsiders in networks.

2.1 Structure-connected cluster

The existing network clustering methods such as modularity based algorithms are designed to find optimal clusters based on the number of edges between vertices or between clusters. Direct connections are important, but they represent only one aspect of the network structure. We think the neighborhood around two connected vertices is also important. The neighborhood of a vertex includes all the vertices connected to it by an edge. When you consider a pair of connected vertices, their combined neighborhood reveals neighbors common to both.

Our method is based on common neighbors. Two vertices are assigned to a cluster according to how they share neighbors. This makes sense when you consider social communities. People who share many friends create a community, and the more friends they have in common, the more intimate the community. But in social networks there are different kinds of actors beside peers. There are also people who are outsider (like hermits), and there are people who are friendly with many communities but belong to none (like politicians). The latter play a special role in small-world networks as *hubs* [10]. An outsider is illustrated by vertex 13 in Figure 1 and a hub is illustrated by vertex 6.

In this paper, we focus on simple, undirected and un-weighted graph. Let $G = \{V, E\}$ be a graph representing a real network, where V is a set of vertices; and E is set of pairs (unordered) of distinct vertices, called edges.

The structure of a vertex can be described by its neighborhood. A formal definition of vertex structure is given as follows.

DEFINITION 1 (VERTEX STRUCTURE)

Let $v \in V$, the structure of v is defined by its neighborhood, denoted by $\Gamma(v)$

$$\Gamma(v) = \{w \in V \mid (v,w) \in E\} \cup \{v\}$$

In Figure 1 vertex 6 is a hub sharing neighbors with two clusters. If we only use the number of shared neighbors, vertex 6 will be clustered into either of the clusters or cause the two clusters to merge. Therefore, we normalize the number of common neighbors in different ways, which give us different similarity measures.

DEFINITION 2 (STRUCTURAL SIMILARITY)

$$\sigma_{\cos}(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}} \quad (2.1)$$

$$\sigma_{jaccard}(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{|\Gamma(v) \cup \Gamma(w)|} \quad (2.2)$$

$$\sigma_{\min}(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\min(|\Gamma(v)|,|\Gamma(w)|)} \quad (2.3)$$

The first similarity, called cosine similarity, normalizes the number of common neighbors by the geometric mean of the two neighborhoods' size and is commonly used for information retrieval. The second similarity, called Jaccard similarity, normalizes the number of common neighbors by the arithmetic mean of the two neighborhoods' size. The third similarity, called min similarity, normalizes the number of common neighbors by the minimum of the two neighborhoods' size. In section 5, we compare the similarities with respect to the clustering accuracy.

When a member of a cluster shares a similar structure with one of its neighbors, their computed structural similarity will be large. We apply a threshold ε to the computed structural similarity when assigning cluster membership, formulized in the following ε-neighborhood definition.

DEFINITION 3 (ε-NEIGHBORHOOD)

$$N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$$

When a vertex shares structural similarity with enough neighbors, it becomes a nucleus or *seed* for a cluster. Such a vertex is called a core vertex. Core vertices are a special class of vertices that have a minimum of μ, neighbors with a structural similarity that exceeds the threshold ε. From core vertices we grow the clusters. In this way the parameters μ and ε determine the clustering of the network. For a given ε, the minimal size of a cluster is determined by μ.

DEFINITION 4 (CORE)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. A vertex $v \in V$ is called a core w.r.t. ε and μ, if its ε-neighborhood contains at least μ vertices, formally:
$CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geq \mu$

We grow clusters from core vertices as follows. If a vertex is in ε-neighborhood of a core, it should be also in the same cluster because they share a similar structure and are connected. This idea is formulized in the following definition of direct structure reachability.

DEFINITION 5 (DIRECT STRUCTURE REACHABILITY)

$$DirREACH_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge w \in N_\varepsilon(v)$$

Direct structure reachablility is symmetric for any pair of cores. However, it is asymmetric if one of the vertices is not a core. The following definition is a canonical extension of direct structure reachability.

DEFINITION 6 (STRUCTURE REACHABILITY)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. A vertex $w \in V$ is structure reachable from $v \in V$ w.r.t ε and μ, if there is a chain of vertices $v_1,\ldots,v_n \in V$, $v_1 = v$, $v_n = w$ such that v_{i+1} is directly structure reachable from v_i, formally:

$$REACH_{\varepsilon,\mu}(v,w) \Leftrightarrow$$
$$\exists v_1,\ldots v_n \in V : v_1 = v \wedge v_n = w \wedge$$
$$\forall i \in \{1,\ldots,n-1\} : DirREACH_{\varepsilon,\mu}(v_i, v_{i+1}).$$

The structure reachability is transitive, but it is asymmetric. It is only symmetric for a pair of cores. More specifically, the structure-reachability is a transitive closure of direct structure-reachablility.

Two non-core vertices in the same cluster may not be structure-reachable because the core condition may not hold for them. But they still belong to the same cluster because they both are structure reachable from the same core. This idea is formulized in the following definition of structure connectivity.

DEFINITION 7 (STRUCTURE CONNECTIVITY)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. A vertex $v \in V$ is structure-connected to a vertex $w \in V$ w.r.t ε and μ, if there is a vertex $u \in V$ such that both v and w are structure reachable from u, formally:

$$CONNECT_{\varepsilon,\mu}(v,w) \Leftrightarrow$$
$$\exists u \in V : REACH_{\varepsilon,\mu}(u,v) \wedge REACH_{\varepsilon,\mu}(u,w).$$

The structure connectivity is a symmetric relation. For the structure reachable vertices, it is also reflective.

Now we are ready to define a cluster as structure-connected vertices, which is maximal w.r.t. structure reachability.

DEFINITION 8 (STRUCTURE-CONNECTED CLUSTER)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. A non-empty subset $C \subseteq V$ is called a structure-connected cluster (SCC) w.r.t ε and μ, if all vertices in C are structure-connected and C is maximal w.r.t structure reachability, formally:

$$SCC_{\varepsilon,\mu}(C) \Leftrightarrow$$

(1) Connectivity:
$$\forall v,w \in C : CONNECT_{\varepsilon,\mu}(v,w)$$

(2) Maximality:
$$\forall v,w \in V : v \in C \wedge REACH_{\varepsilon,\mu}(v,w) \Rightarrow w \in C$$

Now we can define a clustering of a network G w.r.t. the given parameters ε and μ as all structure-connected clusters in G.

DEFINITION 9 (CLUSTERING)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. A clustering P of network $G = <V, E>$ w.r.t. ε and μ consists of all structure-connected clusters w.r.t. ε and μ in G, formally:

$$CLUSTERING_{\varepsilon,\mu}(P) \Leftrightarrow P = \{C \subseteq V \mid SCC_{\varepsilon,\mu}(C)\}$$

Based on the clustering definition above, some vertices may not belong to any clusters. They are outliers in the sense that structurally they are not similar to their neighbors, formally:

DEFINITION 10 (OUTLIER)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. For a given clustering P, i.e. $CLUSTERING_{\varepsilon,\mu}(P)$, if a vertex $v \in V$ does not belong to any clusters, it is an outlier w.r.t. ε and μ, formally,

$$OUTLIER_{\varepsilon,\mu}(v) \Leftrightarrow \forall C \in P: v \notin C$$

The outliers may play different rolls. Some outliers, such as vertex 6 in Figure 1, connect to many clusters and act as a hub. Others, such as vertex 13 in Figure 1, connect to relatively few clusters and are potentially outsiders because they have only week connections to the network. In the following, we formulize the notion of an outlier and their classification as either hubs or outsiders.

DEFINITION 11 (HUB)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. For a given clustering P, i.e. $CLUSTERING_{\varepsilon,\mu}(P)$, if an outlier $v \in V$ has neighbors belonging to two or more different clusters w.r.t. ε and μ, it is a hub (it bridges different clusters) w.r.t. ε and μ, formally,

$HUB_{\varepsilon,\mu}(v) \Leftrightarrow$

(1) $OUTLIER_{\varepsilon,\mu}(v)$

(2) v bridges different clusters:

$$\exists p, q \in \Gamma(v): \exists X, Y \in P: X \neq Y \wedge p \in X \wedge q \in Y.$$

DEFINITION 12 (OUTSIDER)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. For a given clustering P, i.e. $CLUSTERING_{\varepsilon,\mu}(P)$, an outlier $v \in V$ is an outsider if and only if all its neighbors belong to a single cluster or other outliers, formally,

$OUTSIDER_{\varepsilon,\mu}(v) \Leftrightarrow$

(1) $OUTLIER_{\varepsilon,\mu}(v)$

(2) v does not bridge different clusters:

$$\neg\exists p, q \in \Gamma(v): \exists X, Y \in P: X \neq Y \wedge p \in X \wedge q \in Y.$$

In practice, the definition of a hub and an outsider is flexible. The more clusters an outlier bridges, the more strongly that vertex is indicated to be a hub. This point will be discussed further when we consider actual networks.

The following lemmas are important for validating the correctness of our proposed algorithm. Intuitively, the lemmas mean the following. Given a network $G=<V,E>$ and two parameters ε and μ, we can find structure-connected clusters in a two-step approach. First, choose an arbitrary vertex from V satisfying the core condition as a seed. Second, retrieve all the vertices that are structure reachable from the seed to obtain the cluster grown from the seed.

LEMMA 1.

Let $v \in V$. If v is a core, then the set of vertices, which are structure reachable from v is a structure connected cluster, formally:

$$CORE_{\varepsilon,\mu}(v) \wedge C = \{w \in V \mid REACH_{\varepsilon,\mu}(v,w)\}$$

$$\Rightarrow SCC_{\varepsilon,\mu}(C)$$

PROOF:

(1) $C \neq 0$:

By assumption, $CORE_{\varepsilon,\mu}(v)$ and thus, $RECH_{\varepsilon,\mu}(v,v) \Rightarrow v \in C$.

(2) Maximality:

Let $p \in C$ and $q \in V$ and $RECH_{\varepsilon,\mu}(p,q)$.

$\Rightarrow RECH_{\varepsilon,\mu}(v,p) \wedge RECH_{\varepsilon,\mu}(p,q)$

$\Rightarrow RECH_{\varepsilon,\mu}(v,q)$, since structure reachability is transitive.

$\Rightarrow q \in C$.

(3) Connectivity:

$\forall\, p, q \in C$: $RECH_{\varepsilon,\mu}(v,p) \wedge RECH_{\varepsilon,\mu}(v,q)$

$\Rightarrow CONNECT_{\varepsilon,\mu}(p,q)$, via v. □

Furthermore, a structure-connected cluster C with respect to ε, μ is uniquely determined by *any* of its cores, i.e., each vertex in C is structure reachable from any of the cores of C and, therefore, a structure-connected cluster C contains exactly the vertices which are structure reachable from an arbitrary core of C.

LEMMA 2.

Let $C \subseteq V$ be a structure-connected cluster. Let $p \in C$ be a core. Then, C equals the set of vertices, which are structure reachable from p, formally:

$$SCC_{\varepsilon,\mu}(C) \wedge p \in C \wedge CORE_{\varepsilon,\mu}(p)$$

$$\Rightarrow C = \{v \in V \mid RECH_{\varepsilon,\mu}(p,v)\}$$

PROOF:

Let $\hat{C} = \{v \in V \mid RECH_{\varepsilon,\mu}(p,v)\}$. We have to show that $C = \hat{C}$:

(1) $\hat{C} \subseteq C$: it is obvious from the definition of $\hat{C}$.

(2) $C \subseteq \hat{C}$: Let $q \in C$. By assumption, $p \in C \wedge SCC_{\varepsilon,\mu}(C)$.

$\Rightarrow \exists\, u \in C$: $RECH_{\varepsilon,\mu}(u,p) \wedge RECH_{\varepsilon,\mu}(u,q)$

$\Rightarrow RECH_{\varepsilon,\mu}(p,u)$, since both u and p are cores; and structure reachability is symmetric for cores.

$\Rightarrow RECH_{\varepsilon,\mu}(p,q)$, since structure reachability is transitive.

$\Rightarrow q \in \hat{C}$. □

3. ALGORITHM

In this section, we describe the algorithm which implements the search for clusters, hubs and outsiders. As mentioned in section 2.1, the search visiting each vertex once to find structure-connected clusters and the outliers, and then classifies each outlier as either a hub or an outsider based on their connectivity to the clusters.

The pseudo code of the algorithm is presented in Figure 2. The algorithm performs one pass of a network and finds all structure-connected clusters for a given parameter setting. At the beginning all vertices are labeled as unclassified. The algorithm either assigns a vertex to a cluster or labels it as an outlier. For each vertex that is not yet classified, it checks whether this vertex is a core (STEP 1 in Figure 2). If the vertex is a core, a new cluster is expanded from this vertex (STEP 2.1 in Figure 2). Otherwise, the vertex is labeled as an outlier (STEP 2.2 in Figure 2). To find a new cluster, the algorithm starts with an arbitrary core v and search for all vertices that are structure-reachable from v in STEP 2.1. This is sufficient to find the complete cluster containing vertex v, due to lemma 2. In STEP 2.1, a new cluster ID is generated that will be assigned to all vertices found in STEP 2.1. The algorithm begins by inserting all vertices in ε-neighborhood of vertex v into a queue. For each vertex in the queue it computes all directly reachable vertices and inserts those vertices into the queue which are still unclassified. This is repeated until the queue is empty.

```
ALGORITHM (G=<V, E>, ε, μ)
// all vertices in V are labeled as unclassified;
for each unclassified vertex v ∈ V do
// STEP 1. check whether v is a core;
    if CORE_{ε,μ}(v) then
// STEP 2.1. if v is a core, a new cluster is expanded;
        generate new cluster ID;
        insert all x ∈ N_ε(v) into queue Q;
        while Q ≠ 0 do
            y = first vertex in Q;
            R = {x ∈ V | DirRECH_{ε,μ}(y, x)};
            for each x ∈ R do
                if x is unclassified or an outlier then
                    assign current cluster ID to x;
                if x is unclassified then
                    insert x into queue Q;
            remove y from Q;
    else
// STEP 2.2. if v is not a core, it is labeled as an outlier
        label v as outlier;
end for.
// STEP 3. further classifies outliers
for each outlier v do
    if ( ∃ x, y ∈ Γ(v) ( x.clusterID ≠ y.clusterID then
      label v as hub
    else
      label v as outsider;
end for.
end.
```

Figure 2. The Pseudo Code of our Algorithm

The outliers can be further classified as hubs or outsiders in STEP 3. If an outlier connects to two or more clusters, it is classified as a hub. Otherwise, it is an outsider. This final classification is done according to what is appropriate for the network. As mentioned earlier, the more clusters in which an outlier has neighbors, the more strongly that vertex acts as a hub between those clusters. Likewise, a vertex might bridge only two clusters,

but how strongly it is viewed as a hub may depend on how aggressively it bridges them.

As discussed in Section 2, the results of our algorithm do not depend on the order the vertices are processed. The partitioning (number of clusters and association of cores to clusters) is determinate.

4. COMPLEXITY ANALYSIS

In this section, we present an analysis of the computation complexity of the algorithm. Given a network with m edges and n vertices, we first find all structure-connected clusters w.r.t. a given parameter setting by checking each vertex of the network (STEP 1 in Figure 2). This entails retrieval of all the vertex's neighbors. Using an adjacency list, a data structure where each vertex has a list of which vertices it is adjacent to, the cost of a neighborhood query is proportional to the number of neighbors, that is, the degree of the query vertex. Therefore, the total cost is $O(\deg(v_1)+\deg(v_2)+\ldots\deg(v_n))$, where $\deg(v_i)$, $i = 1,2,\ldots,n$ is the degree of vertex v_i. If we sum all the vertex degrees in G, we count each edge exactly twice: once from each end. Thus the running time is $O(m)$.

We also derive the running time in terms of the number of vertices, should the number of edges be unknown. In the worst case, each vertex connects to all the other vertices for a complete graph. The worst case total cost, in terms of the number of vertices, is $O(n(n\text{-}1))$, or $O(n^2)$. However, real networks generally have sparser degree distributions. In the following we derive the complexity for an average case, for which we know the probability distribution of the degrees. One type of network is the random graph, studied by Erdös and Rényi [17]. Random graphs are generated by placing edges randomly between vertices. Random graphs have been employed extensively as models of real world networks of various types, particularly in epidemiology. The degree of a random graph has a Poisson distribution:

$$p(k) = \binom{n}{k} p^k (1-p)^{n-k} \approx \frac{z^k e^z}{k!},$$

which indicates that most nodes have approximately the same number of links (close to the average degree $E(k)=z$). In the case of random graphs the complexity of the algorithm is $O(n)$.

Many real networks, such as social networks, biological networks and the WWW follow a power-law degree distribution. The probability that a node has k edges, $P(k)$, is on the order $k^{-\alpha}$, where α is the degree exponent. A value between 2 and 3 was observed for the degree exponent for most biological and non-biological networks studied by the Faloutsos brothers [18] and Barabási and Oltvai [19]. The expected value of degree is $E(k)= \alpha/(\alpha\text{-}1)$. In this case the average cost of the algorithm is again $O(n)$.

We conclude that the complexity in terms of the number of edges in the network for our algorithm is in general linear. The complexity in terms of the number of vertices is quadratic in the worst case of a complete graph. For real networks like social networks, biological networks and computer networks we expect linear complexity with respect to the number of vertices. This is confirmed by our empirical study described in the next section.

5. EVALUATION

In this section we evaluate our algorithm using both synthetic and real datasets. We first compared the different structural similarities defined in section 2.1 for the accuracy of the clustering. The performance of the algorithm is then compared with fast modularity-based network clustering algorithm proposed by Clauset *et al* in [9], which is faster than many competing algorithms: its running time on a network with n vertices and m edges is $O(md \log n)$ where d is the depth of the dendrogram describing the hierarchical cluster structure. We implemented our algorithm in C++. We used the original source code of fast modularity algorithm by Clauset *et al* [14]. All the experiments were conducted on a PC with a 2.0 GHz Pentium 4 processor and 1 GB of RAM.

5.1 Efficiency

To evaluate the computational efficiency of the proposed algorithm we generate ten networks with the number of vertices ranging from 1,000 to 1,000,000 and the number of edges ranging from 2,182 to 2,000,190. We adapted the construction as used in [8] as follows: first we generate clusters such that each vertex connects to vertices within the same cluster with a probability P_i, and connects to vertices outside its cluster with a probability $P_o<P_i$. Next we add a number of hubs and outsiders. An example of a generated network is presented in Figure 3.

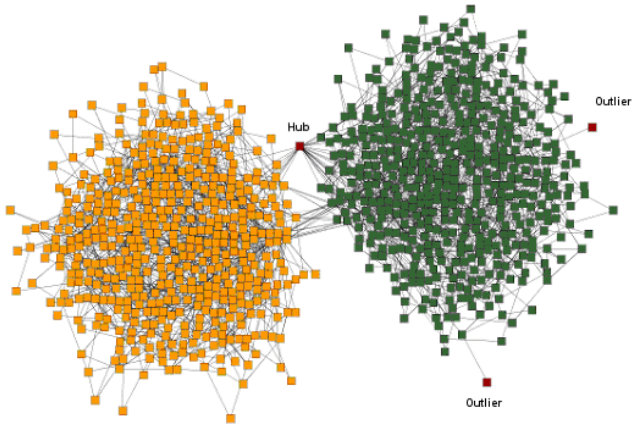


Figure 3. A Synthetic Network with 1,000 Vertices

The running time for fast modularity and our algorithm on the synthetic networks are plotted in Figure 4 and 5, respectively. Running time is plotted both as a function of the number of nodes and the number of edges. Figure 5 shows that our algorithm's running time is in fact linear w.r.t. to the number of vertices and the number of edges, while fast modularity's running time is basically quadratic and scales poorly for large networks. Note the difference in scale for the y-axis between the two figures.

5.2 Effectiveness

To evaluate the effectiveness of network clustering, we use real datasets whose clusters are known a priori. These real datasets include American College Football and Books about US politics. We also apply the clustering algorithm to customer data integration. We use adjusted Rand index (ARI) [12] as a measure of effectiveness of network clustering algorithms in addition to visually comparing the generated clusters to the actual.

5.2.1 Adjusted Rand Index

A measure of agreement is needed when comparing the results of a network clustering algorithm to the expected clustering. Rand Index [11] serves this purpose. One problem with the Rand Index

is that the expected value when comparing two random clusters is not constant. An Adjusted Rand Index (ARI) was proposed by Hubert and Arabie [12] to fix this problem. The ARI is defined as follows:

$$\frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}$$

Where $n_{i,j}$ is the number of vertices in both cluster x_i and y_j; and $n_{i,\cdot}$ and $n_{\cdot,j}$ is the number of vertices in cluster x_i and y_j respectively.

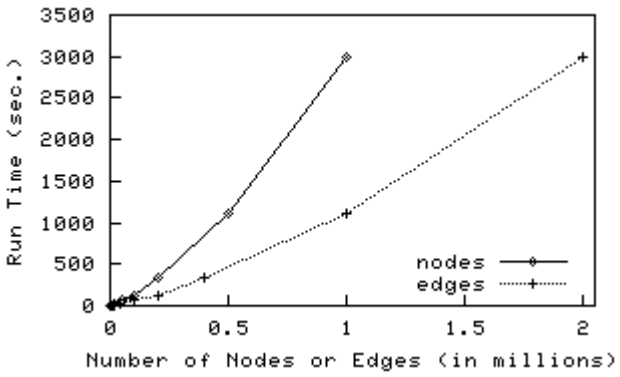


Figure 4. Running Time for fast modularity algorithm

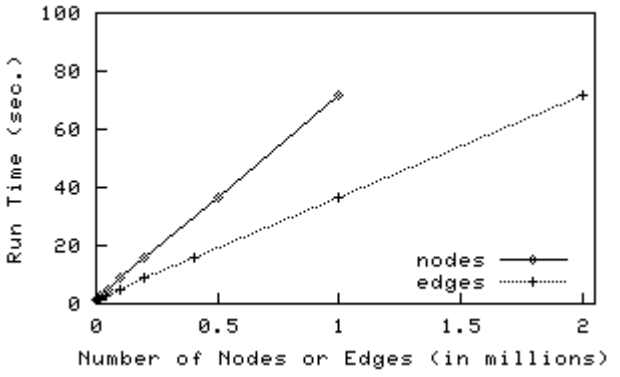


Figure 5. Running Time for our algorithm

Milligan and Cooper [13] evaluated many different indices for measuring agreement between two network clustering with different numbers of clusters and recommend the ARI as the measure of choice. We adopt the ARI as our measure of agreement between the network clustering result and the true clustering of the network. The ARI lies between 0 and 1. When the two clustering agree perfectly, the ARI is 1.

5.2.2 Performance of Various Structural Similarities

Our algorithm groups vertices based on their structural similarity. We compared the various structural similarities defined in section 2.1 for the accuracy of the clusters they generate, measured by ARI. The results on real networks including college football, political books and customers described in sections 5.2.3, 5.2.4 and 5.2.5 respectively are listed in Table 1.

Table 1. Performance of Various Structural Similarities

	COSINE (σ_{cos})	MIN (σ_{min})	JACCARD ($\sigma_{jaccard}$)
College football	1	0.255	0.983
Political books	0.708	0.661	0.574
CG1	1	1	1
CG2	1	0.942	1

The cosine similarity (σ_{cos}) achieves overall the best accuracy in comparison with other similarity measures. In our following experiments, we used cosine similarity for our algorithm.

5.2.3 College Football

The first real dataset we examine is the 2006 NCAA Football Bowl Subdivision (formerly Division 1-A) football schedule. This example is inspired by the set studied by Newman and Girvan [8], who consider contests between Div. 1-A teams in 2000. Our set is more complex, considering all contests of the Bowl Subdivision schools including those against schools in lower divisions.

The challenge is to discover the underlying structure of this network – the college conference system. The National Collegiate Athletic Association (NCAA) divides 115 schools into eleven conferences. In addition there are four independent schools at this top level: Army, Navy, Temple, and Notre Dame. Each Bowl Subdivision school plays against schools within their own conference, against schools in other conferences, and against lower division schools. The network contains 180 vertices (119 Bowl Subdivision schools and 61 lower division schools) interconnected by 787 edges. Figure 6 shows this network with schools in the same conference identified by color.

This example illustrates kinds of structures that our method seeks to address. Schools in the same conference are clusters. The four independent schools play teams in many conferences but belong to none; they are hubs. The lower division schools are only weakly connected to the clusters in the network; they are outsiders.

First we cluster this network by using the fast modularity algorithm. The results, for which the modularity is 0.599 is shown in Figure 7. Maximizing Newman's modularity gives a satisfying network clustering, identifying nine clusters. All schools in the same conference are clustered together. However, two of the conferences are merged (the Western Athletic and Mountain West conferences and the Mid-American and Big Ten conferences), the four independent schools are classified into various conferences despite their hub-like properties. All lower division teams are assigned to clusters.

Next we cluster the network using our algorithm, using the parameters (ε= 0.5, μ = 2). This clustering succeeds in capturing all the features of the network. Eleven clusters are identified, corresponding exactly to the eleven conferences. All schools in the same conference are clustered together. The independent schools and the lower division schools are unclassified – they stand apart from the clusters. The four independent schools show strong properties as hubs; they have inactive edges that connect them to a large number of clusters – at minimum five. In contrast the lower division schools have only week connections to clusters – one or two, and in a single case three. They are true outsiders.

This clustering matches perfectly the underlying structure shown in Figure 6.

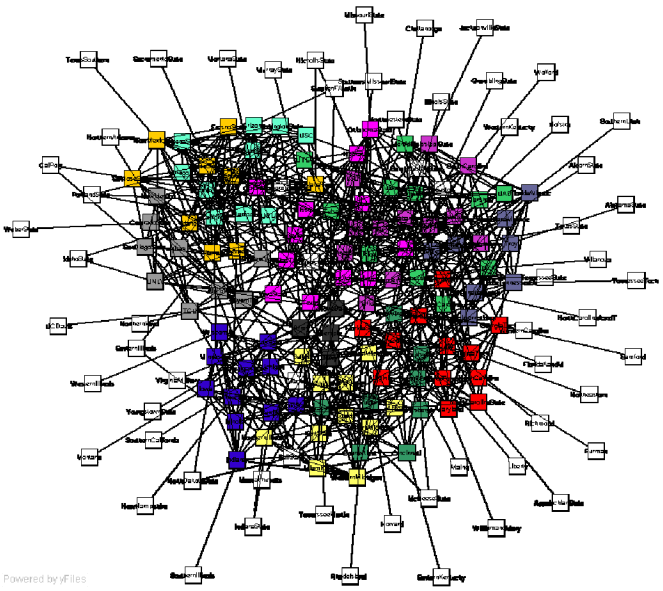


Figure 6. NCAA Football Bowl Subdivision schedule as a network, showing the 12 conferences in color, independent schools in black, and lower division schools in white.

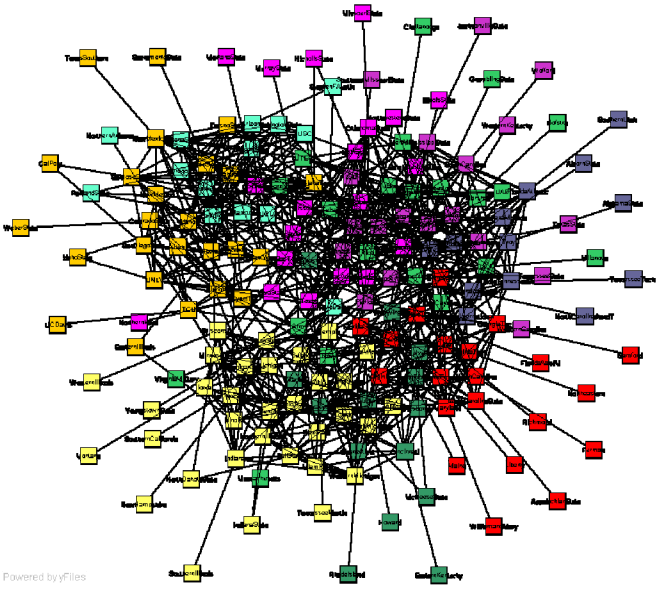


Figure 7. NCAA Football Bowl Subdivision schedule as clustered by fast modularity algorithm.

5.2.4 Books about US politics

The second example is the classification of books about US politics. We use the dataset of Books about US politics compiled by Valdis Krebs [15]. The vertices represent books about US politics sold by the online bookseller Amazon.com. The edges represent frequent co-purchasing of books by the same buyers, as indicated by the "customers who bought this book also bought these other books" feature on Amazon. The vertices have been given values "l", "n", or "c" to indicate whether they are "liberal", "neutral", or "conservative". These alignments were assigned separately by Mark Newman [16] based on a reading of the descriptions and reviews of the books posted on Amazon. The political books network is illustrated in Figure 8. The "conservative", "neutral" and "liberal" books are represented by red, gray and blue respectively.

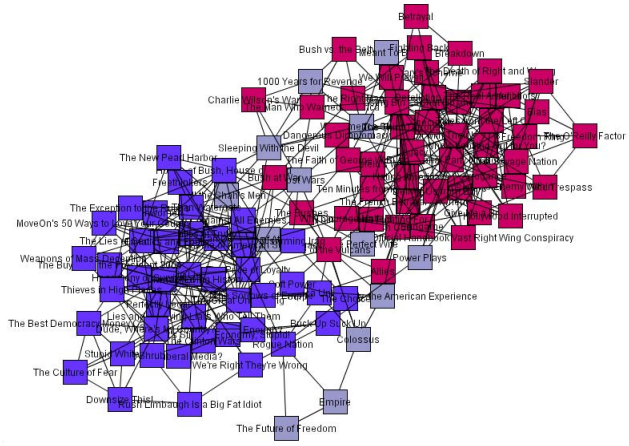


Figure 8. Political Book Network.

First we apply our algorithm to the political books network, using the parameters (ε= 0.35, μ = 2). Our goal is to find clusters that represent the different political orientations of the books. The result is presented in Figure 9. Our algorithm successfully find three clusters representing "conservative", "neutral" and "liberal" books respectively. The obtained clusters are illustrated using three different shapes: squares for "conservative" books and triangles for "neutral" books and circles for "liberal" books. Additionally, each vertex is labeled with the book title.

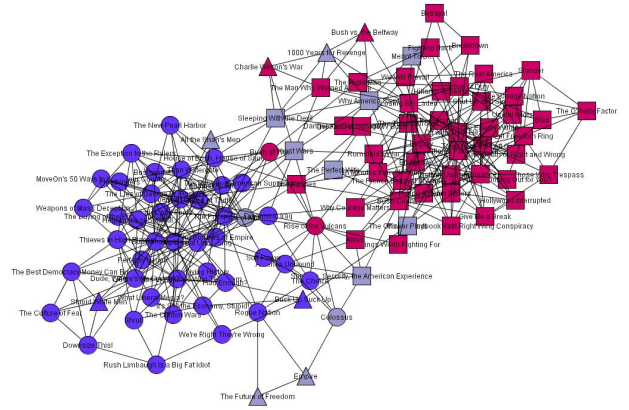


Figure 9. The Result of our algorithm on Political Book Network.

The result for the fast modularity algorithm is presented in Figure 10. The fast modularity algorithm found 4 clusters, presented using circles, triangles, squares, and hexagons. Although two dominant clusters, represented by circles and squares, align well with the "conservative" and "liberal" classes, the "neutral" class is mostly misclassified. This demonstrates again that fast modularity algorithm can not handle vertices that bridge clusters.

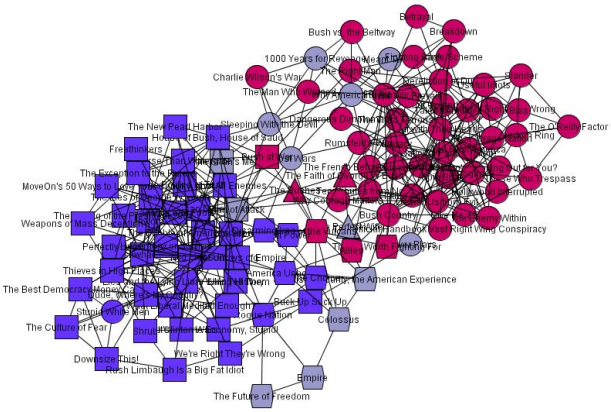
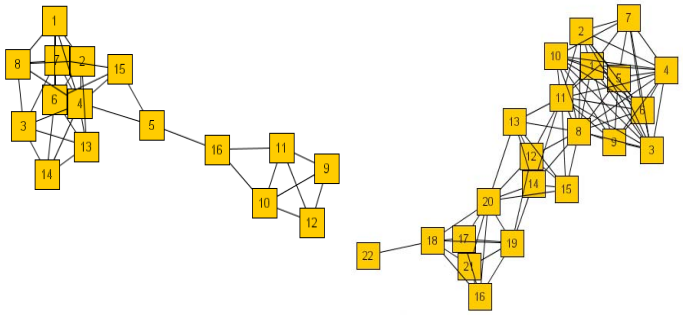


Figure 10. The Result of fast modularity algorithm on Political Book Network.

5.2.5 Customer Data Integration

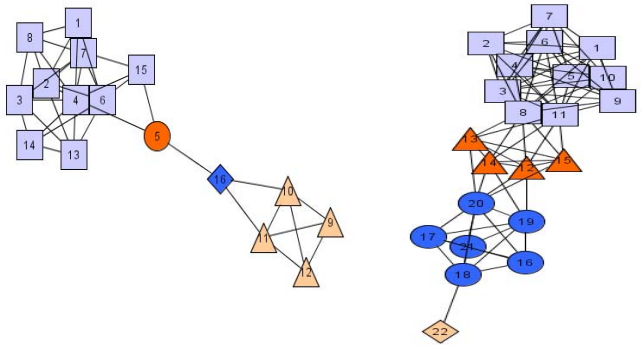
Finally we apply network the clustering algorithms to detecting groups of records for the same individual, a problem called Customer Data Integration (CDI). A large database of records consisting of names and addresses are matched against each other. The database contains multiple records for the same individual, but they manifest variations in the names and addresses whose causes range from the use of nicknames and abbreviations to data entry errors. If two records match we connect them with an edge. From a large file we extract sets of interconnected records for study. We test two networks, CG1 and CG2, (shown in Figure 11). Network CG1 represents data for two individuals and two poor-quality records that represent no true individual. Network CG2 represents four individuals, one of whom is represented by a single instance.



CG1 CG2

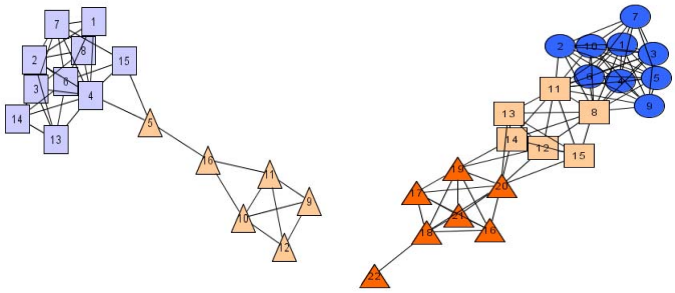
Figure 11. Customer Networks CG1 and CG2.

The clustering results of our algorithm, using the parameters (ε= 0.7, μ = 2), are presented in Figure 12. The results demonstrate that it successfully found all the clusters and outliers. The results of fast modularity algorithm are presented in Figure 13. It is clear that it failed to identify any outliers.



CG1 CG2

Figure 12. The Result of our algorithm on CG1 and CG2.



CG1 CG2

Figure 13. The Result of fast modularity algorithm on CG1 and CG2.

5.2.6 Adjusted Rand Index Comparison

As mentioned in Section 5.2.1, the Adjust Rand Index is an effective measure of the similarity of a clustering result to the true clustering. The results for College Football, Political Books, CG1 and CG2 are presented in Table 2.

Table 2. Adjust Rand Index Comparison.

	Our algorithm	Fast modularity algorithm
College football	**1**	0.24
Political books	**0.71**	0.64
CG1	**1**	0.85
CG2	**1**	0.68

The ARI results clearly demonstrate that the proposed algorithm outperforms fast modularity algorithm at producing clustering that resemble the true clustering for the real world networks in our study.

5.2.7 Input Parameters

Our algorithm uses two parameters: ε and μ. To choose them we adapted the heuristic suggested for DBSCAN in [7]. This involves making a *k*-nearest neighbor query for a sample of vertices and noting the nearest structural similarity as defined in Section 2.1. The query vertices are then sorted in ascending order of nearest structural similarity. A typical *k*-nearest similarity plot is shown in Figure 14. The knee indicated by a vertical line shows that an appropriate ε value for this network is 0.7. This knee represents a

separation of vertices belonging to clusters to the right from hubs and outliers to the left. Usually a sample of 10% of the vertices is sufficient to locate the knee. In the absence of such an analysis, an ε value between 0.5 and 0.8 is normally sufficient to achieve a good clustering result. We recommend a value of 2 for μ.

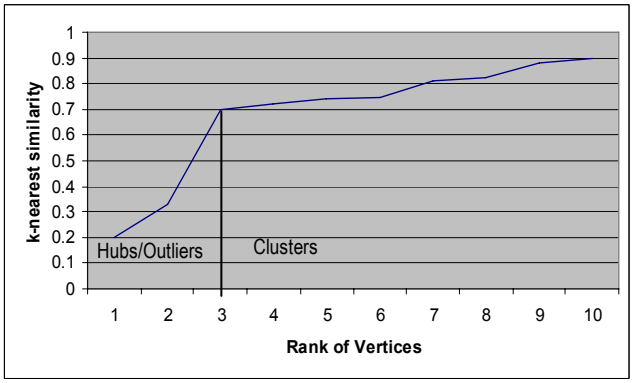


Figure 14. Sorted k-Nearest Structural Similarity.

6. RELATED WORK

Network clustering is the division of a network into set of sub-networks, called clusters. More specifically, given a network $G = \{V, E\}$, where V is a set of vertices and E is a set of edges between vertices, the goal of network clustering is to divide G into k disjoint sub-networks $G_i = \{V_i, E_i\}$, in which $V_i \cap V_j = \Phi$ for any $i \neq j$, and $V = \sum_{i=1}^{k} V_i$. The number of sub-networks, k, may or may not be known a priori. In this paper, we focus on simple, undirected, and un-weighted networks.

The problem of finding good clusters for networks has been studied for some decades in many fields, particularly computer science and physics. Here we review some of the more common methods.

The ***min-max cut method*** [1] seeks to cluster a network G=$\{V, E\}$ into two clusters A and B. The principle of min-max clustering is minimizing the number of connections between A and B and maximizing the number of connections within each. A ***cut*** is defined the number of edges that would have to be removed to isolate the vertices in cluster A from those in cluster B. The min-max cut algorithm searches for the clustering that creates two clusters whose *cut* is minimized and while maximizing the number of remaining edges.

A pitfall of this method is that, if one cuts out a single vertex from the network, one will probably achieve the optimum. Therefore, in practice, the optimization must be accompanied with some constraint, such as A and B should be of equal or similar size, or $|A| \approx |B|$. Such constraints are not always appropriate; for example, in social networks some communities are much larger than the others.

To amend the issue, a ***normalized cut*** was proposed [2], which normalizes the cut by the total number connections between each cluster to the rest of the network. Therefore, cutting out one vertex or some small part of the network will no longer always yield an optimum.

Both min-max cut and normalized cut methods cluster a network into two clusters. To divide a network into k clusters, one has to adopt a top-down approach, splitting the network into two clusters, and then further splitting these clusters, and so on, until k clusters have been detected. There is no guarantee of the optimality of recursive clustering. There is no measure of the number of clusters that should be produced when k is unknown. There is no indicator to stop the bisection procedure.

Recently, ***modularity*** was proposed as a quality measure of network clustering [8]. For a clustering of network with k clusters, the modularity is defined as:

$$Q = \sum_{s=1}^{k} \left[\frac{l_s}{L} - \left(\frac{d_s}{2L} \right)^2 \right]$$

L is the number of edges in the network, l_s is the number of edges between vertices within cluster s, and d_s is the sum of the degrees of the vertices in cluster s. The modularity of a clustering of a network is the fraction of all edges that lie within each cluster minus the fraction that would lie within each cluster if the network's vertices were randomly connected. Optimal clustering is achieved when the modularity is maximized. Modularity is defined such that it is 0 for two extreme cases: when all vertices clustered into a single cluster, and when the vertices are clustered at random. Note that the modularity measures the quality of any network clustering. Normalized and min-max cut measures only the quality of a clustering of two clusters.

Finding the maximum Q is NP-complete. Instead of performing an exhaustive search, various optimization approaches are proposed. For example, a greedy method based on a hierarchical agglomeration clustering algorithm is proposed in [9], which is faster than many competing algorithms: its running time on a network with n vertices and m edges is $O(md \log n)$ where d is the depth of the dendrogram describing the hierarchical cluster structure. Also, Guimera and Amaral [3] optimize modularity using simulated annealing.

To summarize, the network clustering methods discussed in this section aim to find clusters such that there are many connections between vertices within the same clusters and few without. While all these network clustering methods successfully find clusters, they are generally unable to detect hubs and outsiders like those in the example in Figure 1. Such vertices invariably are included in one cluster or another.

7. CONCLUSIONS

Organizing related data is a fundamental task in many fields of science and engineering. Many algorithms for partitioning networks have been proposed from practitioners in different disciplines including computer science and physics. Successful examples are Min-Max Cut [1] and Normalized Cut [2], as well as Modularity-based network clustering algorithms [3][8][9]. While such algorithms can successfully detect clusters in networks, they tend to fail to identify and isolate two kinds of vertices that play special roles – vertices that bridge clusters (hubs) and vertices that are marginally connected to clusters (outsiders). Identifying hubs is particularly valuable for applications such as viral marketing and epidemiology. As vertices that bridge clusters, hubs are responsible for spreading ideas or disease. In contrast, outsiders have little or no influence, and may be isolated as noise in the data.

In this paper, we proposed a new algorithm to detect clusters, hubs and outsiders in networks. It clusters vertices based on their

common neighbors. Two vertices are assigned to cluster according to how they share neighbors. This makes sense when you consider social communities. People who share many friends create a community, and the more friends they have in common, the more intimate the community. But in social networks there are different kinds of actors. There are also people who are outsider (like hermits), and there are people who are friendly with many communities but belong to none (like politicians). The latter play a special role in small-world networks [10].

We applied our algorithm to some real world networks including finding conferences using only the NCCA College Football schedule, grouping political books based on co-purchasing information, and customer data integration. In addition, we compared the new algorithm with the fast modularity-based algorithm in terms of both efficiency and effectiveness. The theoretical analysis and empirical evaluation demonstrate superior performance over the modularity-based network clustering algorithm.

In the future we plan to apply our algorithm to analyze biological networks such as metabolic networks and gene co-expression networks.

8. REFERENCES

[1] C. Ding, X. He, H. Zha, M. Gu, and H. Simon, "A min-max cut algorithm for graph partitioning and data clustering", Proc. of ICDM 2001.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation", IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol 22, No. 8, 2000.

[3] R. Guimera and L. A. N. Amaral, "Functional cartography of complex metabolic networks." Nature 433, 895–900 (2005).

[4] J. Kleinberg. "Authoritative sources in a hyperlinked environment." Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.

[5] P. Domingos and M. Richardson, "Mining the Network Value of Customers", Proc. 7th ACM SIGKDD, pp. 57 – 66, 2001.

[6] Y. Wang, D. Chakrabarti, C. Wang and C. Faloutsos, "Epidemic Spreading in Real Networks: An Eigenvalue Viewpoint", SRDS 2003 (pages 25-34), Florence, Italy

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR, pages 291-316. AAAI Press, 1996.

[8] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks", Phys. Rev. E 69, 026113 (2004).

[9] A. Clauset, M. Newman, and C. Moore, "Finding community in very large networks", 2004.

[10] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," Nature, 393:440-442 (1998)

[11] W. M. Rand, "*Objective criteria for the evaluation of clustering methods*." Journal of the American Statistical Association, 66, pp846–850 (1971).

[12] L. Hubert and P. Arabie, "Comparing Partitions". *Journal of Classification*, 193–218, 1985.

[13] G. W. Milligan and M. C. Cooper, "A study of the comparability of external criteria for hierarchical cluster analysis", Multivariate Behavioral Research, 21, 441–458, 1986.

[14] http://cs.unm.edu/~aaron/research/fastmodularity.htm.

[15] http://www.orgnet.com/.

[16] http://www-personal.umich.edu/~mejn/netdata/.

[17] P. Erdös and A. Rényi, Publ. Math. (Debrecen) 6, 290 (1959).

[18] M. Faloutsos, P. Faloutsos and C. Faloutsos, On Power-Law Relationships of the Internet Topology, SIGCOMM 1999.

[19] A.-L. Barabási and Z. N. Oltvai, Nature Reviews Genetics 5, 101-113 (2004).