SCAN: A Structural Clustering Algorithm for Networks

Xiaowei Xu
University of Arkansas at Little Rock
xwxu@ualr.edu

Nurcan Yuruk, Zhidan Feng
University of Arkansas at Little Rock
{nxyuruk, zxfeng@ualr.edu}

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

ABSTRACT

Network clustering (or graph partitioning) is an important task for the discovery of underlying structures in networks. Many algorithms find clusters by maximizing the number of intra-cluster edges. While such algorithms find useful and interesting structures, 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 (outliers). Identifying hubs is useful for applications such as viral marketing and epidemiology since hubs are responsible for spreading ideas or disease. In contrast, outliers have little or no influence, and may be isolated as noise in the data. In this paper, we proposed a novel algorithm called SCAN (Structural Clustering Algorithm for Networks), which detects clusters, hubs and outliers in networks. It clusters vertices based on a structural similarity measure. The algorithm is fast and efficient, visiting each vertex only once. An empirical evaluation of the method using both synthetic and real datasets demonstrates superior performance over other methods such as the modularity-based algorithms.

Categories and Subject Descriptors

I.5.3 [**PATTERN RECOGNITION**]: Clustering – *Algorithms, Similarity measures.*

General Terms

Algorithms, Performance

Keywords

Network clustering, Graph partitioning, Community Structure, Hubs, Outliers

1. INTRODUCTION

Much data of current interest to the scientific community can be modeled as networks (or graphs). A network is sets of vertices, representing objects, connected together by edges, representing the relationship between objects. For example, a social network can be viewed as a graph where individuals are represented by vertices; and the friendship between individuals are edges [1]. Similarly, the world-wide web can be modeled as a graph, where web pages are represented as vertices that are connected by an edge when one pages contains a hyperlink to another [2] [3].

Network clustering (or graph partitioning) is a fundamental approach for detecting hidden structures in networks that, because of many interesting applications, is drawing increased attention in computer science [4][5], physics [11], and bioinformatics [6]. Various methods have been developed. These methods tend to cluster networks such that there are a dense set of edges within every cluster and few edges between clusters. Modularity-based algorithms [6][11][12] and normalized cut [4][5] are successful examples. However, they do not distinguish the roles of the vertices in the networks. Some vertices are members of clusters; some vertices are hubs that bridge many clusters but don't belong to any, and some vertices are outliers that have only a weak association with a particular cluster. The situation is illustrated in Figure 1.

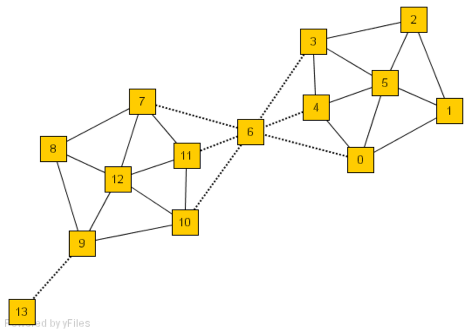

Figure 1. A Network with 2 Clusters, a Hub and an Outlier.

The existing methods such as modularity-based algorithm [12] will partition this example into two clusters: one consisting of vertices 0 to 6 and the other consisting of vertices 7 to 13. They do not isolate vertex 6, a hub whose membership in either cluster is disputable, or vertex 13, which has only a single connection to the network.

The identification and isolation of hubs is essential for many applications. As an example, the identification of hubs in the WWW improves the search for relevant authoritative web pages [7]. Furthermore, hubs are believed to play a crucial role in viral marketing [8] and epidemiology [9].

In this paper, we propose a new method for network clustering called SCAN (Structural Clustering Algorithm for Networks). The goal of our method is to find clusters, hubs, and outliers 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 grouped into the clusters 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.

Refer again to the example in 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 grouped together in the same cluster. In contrast, consider the neighborhoods of vertex 13 and vertex 9. These two vertices are connected, but share only two common neighbors, i.e. {9, 13}. Therefore, it is doubtful that they should be grouped into the same cluster. The situation for vertex 6 is a little different. It has many neighbors, but they are sparsely interconnected.

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

SCAN has the following features:

- It detects clusters, hubs, and outliers by using the structure and the connectivity of the vertices as clustering criteria. Through theoretical analysis and experimental evaluation we will demonstrate that SCAN can find meaningful clusters and identify hubs and outliers in very large networks.
- It is fast. Its running time on a network with n vertices and m edges is $O(m)$. In contrast, the running time of the fast modularity-based algorithm [12], the fastest existing network clustering algorithm, is $O(md \log n)$.

The paper is organized as follows. We review the related work for network clustering algorithms in section 2. We formulize the notion of structure-connected clusters in section 3. We describe the algorithm SCAN in section 4. We give a computation complexity analysis of SCAN in section 5. We compare SCAN to the fast modularity-based network clustering algorithm in section 6. Finally, we present our conclusions and suggest future work in section 7.

2. RELATED WORK

Network clustering (or graph partitioning) is the division of a graph into a set of sub-graphs, called clusters. More specifically, given a graph $G = \{V, E\}$, where V is a set of vertices and E is a set of edges between vertices, the goal of graph partitioning is to divide G into k disjoint sub-graphs $G_i = \{V_i, E_i\}$, 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-graphs, k, may or may not be known a priori. In this paper, we focus on simple, undirected, and un-weighted graphs.

The problem of finding good clustering of 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*** [4] seeks to partition a graph 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 graph, 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 [5], which normalizes the cut by the total number connections between each cluster to the rest of the graph. Therefore, cutting out one vertex or some small part of the graph will no longer always yield an optimum.

Both min-max cut and normalized cut methods partition a graph into two clusters. To divide a graph into k clusters, one has to adopt a top-down approach, splitting the graph 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 [11]. For a clustering of graph 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 graph, 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 graph is the fraction of all edges that lie within each cluster minus the fraction that would lie within each cluster if the graph'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 partitioned 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 [12], which is faster than many competing algorithms: its running time on a graph 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 [6] 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 outliers like those in the example in Figure 1. Such vertices invariably are included in one cluster or another.

3. THE NOTION OF STRUCTURE-CONNECTED CLUSTERS

Our goal is both to cluster networks optimally and to identify and isolate hubs and outliers. 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 [10] and can distinguish good clusters, hubs, and outliers in networks. In section 4, we present, SCAN, an efficient algorithm to find the optimal clustering of networks.

3.1 Structure-connected Clusters

The existing network clustering methods reviewed in section 2 are designed to find optimal clustering of networks 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 vertices.

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. There are also people who are outsiders (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* [13]. Such a hub is illustrated by vertex 6 in Figure 1.

In this paper, we focus on simple, undirected and un-weighted graph. Let $G = \{V, E\}$ be a graph, 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 by the geometric mean of the two neighborhoods' size.

DEFINITION 2 (STRUCTURAL SIMILARITY)

$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

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 networks. 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. 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 w.r.t ε and μ, if all vertices in C are structure-connected and C is maximal w.r.t structure reachability, formally:

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

(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:

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

A vertex is either a member of a structure-connected cluster, or it is isolated, i.e. it does not belong to any of the structure-connected cluster. If a vertex is not a member of any structure-connected clusters, it is either a hub or an outlier, depending on its neighborhood.

DEFINITION 10 (HUB)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. For a given clustering P, i.e. $CLUSTERING_{\varepsilon,\mu}(P)$, if an isolated vertex $v \in V$ 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,

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

(1) v is not a member of any cluster:

$$\forall\, C \in P: v \notin C$$

(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 11 (OUTLIER)

Let $\varepsilon \in \Re$ and $\mu \in \aleph$. For a given clustering P, i.e. $CLUSTERING_{\varepsilon,\mu}(P)$, an isolated vertex $v \in V$ is an outlier if and only if all its neighbors either belong to only one cluster or do not belong to any cluster, formally,

$OUTLIER_{\in,\mu}(v) \Leftrightarrow$

(1) v is not a member of any cluster:

$$\forall\, C \in P: v \notin C$$

(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 outlier is flexible. It may be more useful to regard hubs as a special kind of outlier, since both are isolated vertices. The more clusters in which an outlier has neighbors, the more strongly that vertex acts as a hub between those clusters. 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 graph $G=<V,E>$ 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:

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

PROOF:

(1) $C \neq 0$:

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

(2) Maximality:

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

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

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

$\Rightarrow q \in C.$

(3) Connectivity:

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

$\Rightarrow CONNECT_{\varepsilon,\mu}(p,q)$, 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:

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

PROOF:

Let $\hat{C} = \{v \in V \mid REACH_{\varepsilon,\mu}(p,v)\}$. 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, $p \in C \wedge CLUSTER_{\varepsilon,\mu}(C)$.

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

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

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

$\Rightarrow q \in \hat{C}.$ □

4. ALGORITHM SCAN

In this section, we describe the algorithm SCAN which implements the search for clusters, hubs and outliers. As mentioned in section 3.1, the search begins by first visiting each vertex once to find structure-connected clusters, and then visiting the isolated vertices to identify them as either a hub or an outlier.

The pseudo code of the algorithm SCAN is presented in Figure 2. SCAN 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 SCAN algorithm classifies each vertex either a member of a cluster or a non-member. For each vertex that is not yet classified, SCAN 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 a non-member (STEP 2.2 in Figure 2). To find a new cluster, SCAN 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 which will be assigned to all vertices found in STEP 2.1. SCAN 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 SCAN($G$=<$V$, $E$>, $\varepsilon$, $\mu$)
// all vertices in $V$ are labeled as unclassified;
for each unclassified vertex $v \in V$ do
// STEP 1. check whether $v$ is a core;
    if CORE$_{\varepsilon,\mu}(v)$ then
// STEP 2.1. if $v$ is a core, a new cluster is expanded;
        generate new clusterID;
        insert all $x \in N_\varepsilon(v)$ into queue $Q$;
        while $Q \neq 0$ do
            $y$ = first vertex in $Q$;
            $R = \{x \in V \mid$ DirREACH$_{\varepsilon,\mu}(y, x)\}$;
            for each $x \in R$ do
                if $x$ is unclassified or non-member then
                    assign current clusterID 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 non-member
        label $v$ as non-member;
end for.
// STEP 3. further classifies non-members
for each non-member vertex $v$ do
    if ( $\exists x, y \in \Gamma(v)$ ( $x$.clusterID ≠ $y$.clusterID) then
        label $v$ as hub
    else
        label $v$ as outlier;
end for.
end SCAN.
```

Figure 2. The Pseudo Code of the Algorithm SCAN

The non-member vertices can be further classified as hubs or outliers in STEP 3. If an isolated vertex has edges to two or more clusters, it is may be classified as a hub. Otherwise, it is an outlier. 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 3, the results of SCAN do not depend on the order of processed vertices, i.e. the obtained clustering of network (number of clusters and association of cores to clusters) is determinate.

5. COMPLEXITY ANALYSIS

In this section, we present an analysis of the computation complexity of the algorithm SCAN. Given a graph with m edges and n vertices, SCAN first finds all structure-connected clusters w.r.t. a given parameter setting by checking each vertex of the graph (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 $O(n(n-1))$, 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 [20]. 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 SCAN 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 [21] and Barabási and Oltvai [22]. The expected value of degree is $E(k)= \alpha/(\alpha-1)$. In this case the average cost of SCAN is again $O(n)$.

Therefore, the complexity in terms of the number of edges in the graph for SCAN 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, SCAN expects linear complexity with respect to the number of vertices. This is confirmed by our empirical study described in the next section.

6. EVALUATION

In this section we evaluate the algorithm SCAN using both synthetic and real datasets. The performance of SCAN is compared with FastModularity, a fast modularity-based network clustering algorithm proposed by Clauset *et al* in [12], which is faster than many competing algorithms: its running time on a graph 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 SCAN in C++. We used the original source code of FastModularity by Clauset *et al* [17]. All the experiments were conducted on a PC with a 2.0 GHz Pentium 4 processor and 1 GB of RAM.

6.1 Efficiency

To evaluate the computational efficiency of the proposed algorithm we generate ten graphs 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 [11] 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 outliers. An example of a generated graph is presented in Figure 3.

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

6.2 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 segmentation. We use adjusted Rand index (ARI) [15] as a measure of effectiveness of network clustering algorithms in addition to visually comparing the generated clusters to the actual.

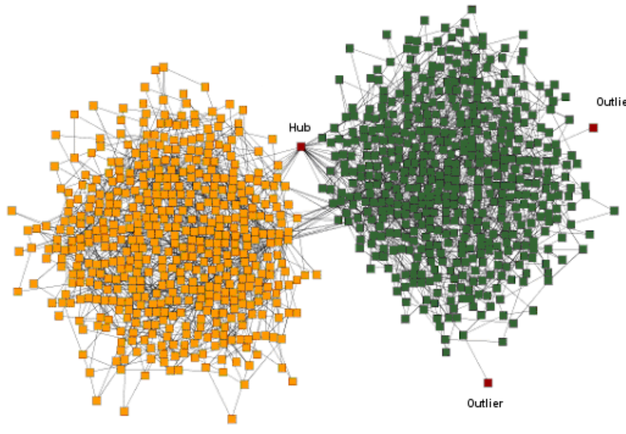

Figure 3. A Synthetic Graph with 1,000 Vertices

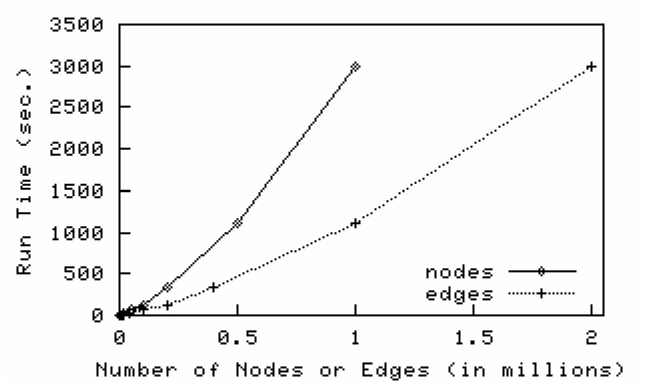

Figure 4. Running Time for FastModularity

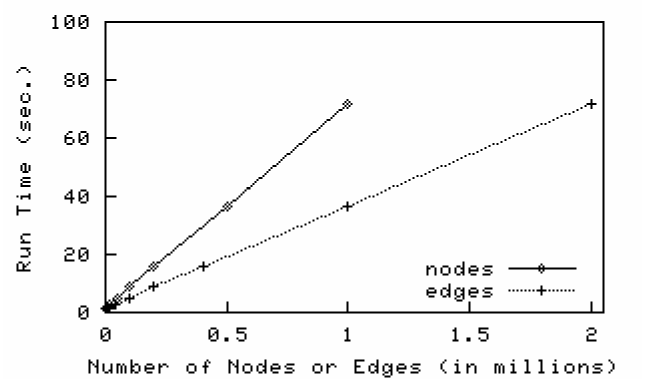

Figure 5. Running Time for SCAN

6.2.1 *Adjusted Rand Index*

A measure of agreement is needed when comparing the results of a network clustering algorithm to the expected clustering. Rand Index [14] serves this purpose. One problem with the Rand Index is that the expected value when comparing two random clustering is not constant. An Adjusted Rand Index was proposed by Hubert and Arabie [15] to fix this problem. The Adjusted Rand Index (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.

Milligan and Cooper [16] evaluated many different indices for measuring agreement between two network clustering with different numbers of clusters and recommend the Adjusted Rand Index as the measure of choice. We adopt the Adjusted Rand Index as our measure of agreement between the network clustering result and the true clustering of the network.

6.2.2 *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 [11], 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.

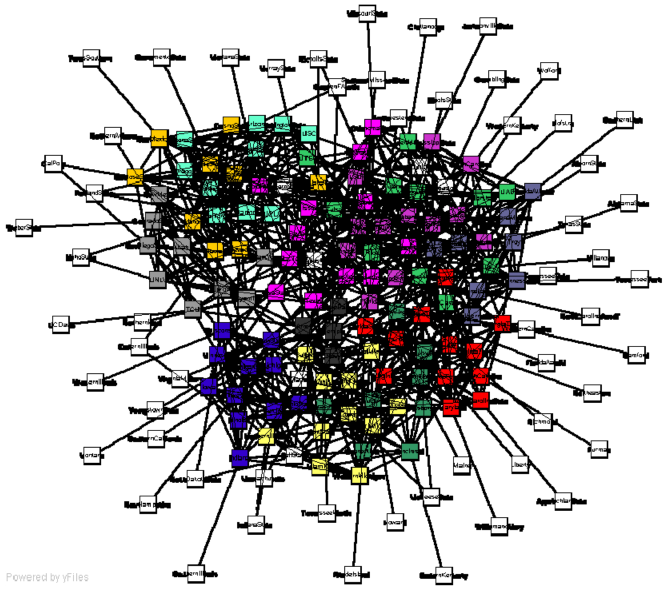

Figure 6. NCAA Football Bowl Subdivision schedule as a network, showing the 12 conferences in color, independent schools in black, and lower division schools in white.

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 outliers.

First we cluster this network by using the FastModularity 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 SCAN algorithm, using the parameters (ε= 0.5, μ = 2). This partition succeeds in capturing all the features of the graph. 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 outliers. This partition matches perfectly the underlying structure shown in Figure 6.

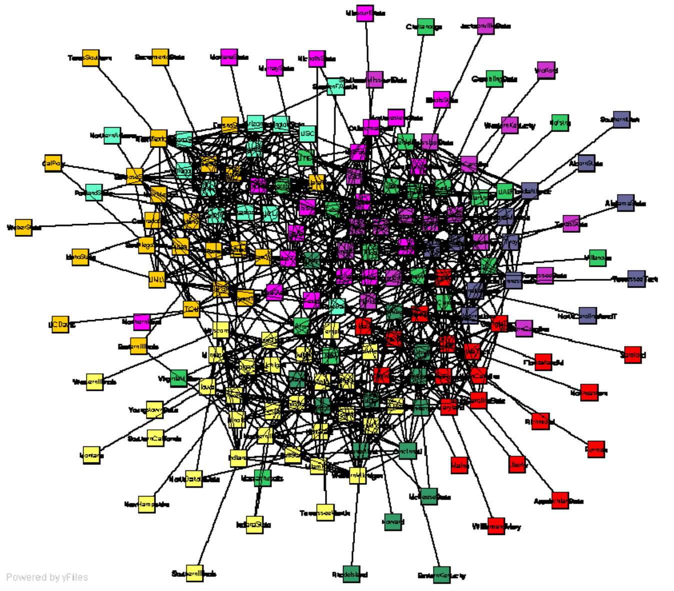

Figure 7. NCAA Football Bowl Subdivision schedule as clustered by FastModularity Algorithm.

6.2.3 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 [18]. 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 [19] based on a reading of the descriptions and reviews of the books posted on Amazon. The political books graph is illustrated in Figure 8. The "conservative", "neutral" and "liberal" books are represented by red, gray and blue respectively.

First we apply the SCAN algorithm to the political books graph, 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. SCAN successfully finds three clusters representing "conservative", "neutral" and "liberal" books respectively. The SCAN 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.

The result for the FastModularity algorithm is presented in Figure 10. FastModularity 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 FastModularity handles poorly vertices that bridge clusters.

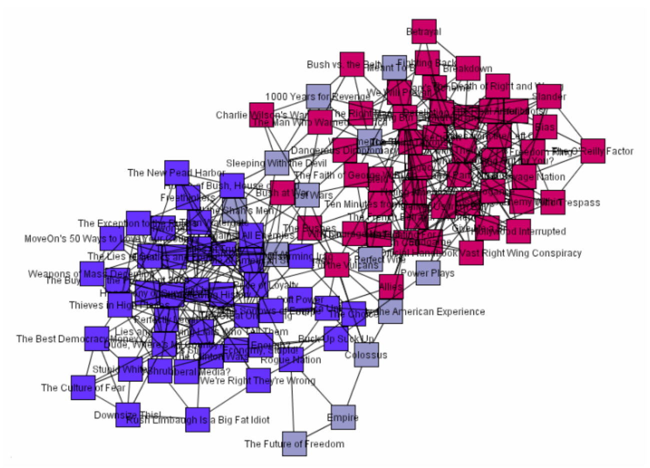

Figure 8. Political Book Graph.

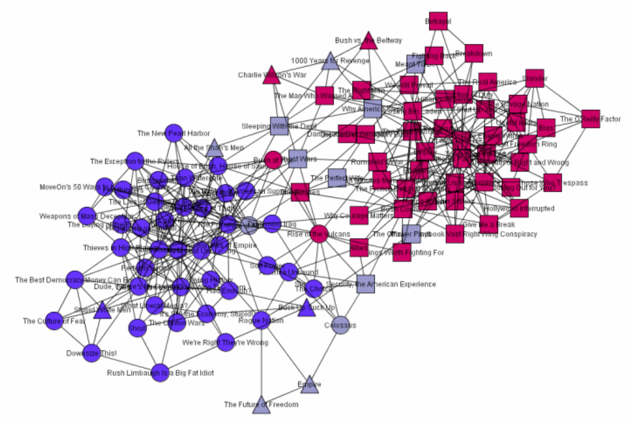

Figure 9. The Result of SCAN on Political Book Graph.

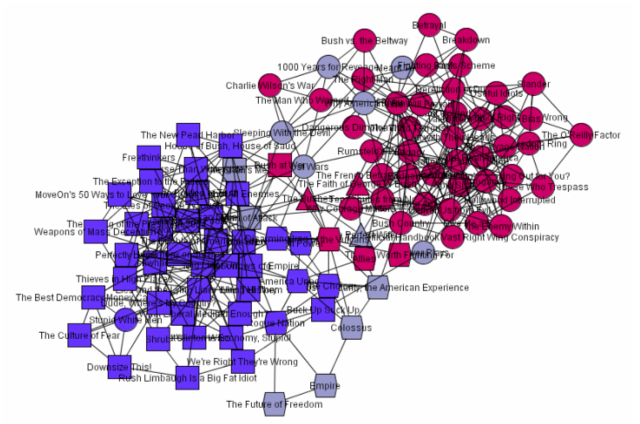

Figure 10. The Result of FastModularity on Political Book Graph.

6.2.4 *Customer Segmentation*

Finally we apply network clustering algorithms to detecting groups of records for the same individual, a problem called Customer Data Integration (CDI). Records consisting of names and addresses are matched against each other using techniques that record with similar information despite variations in the names and addresses. 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 graphs, CG1 and CG2, (shown in Figure 11). Graph CG1 represents data for two individuals and two poor-quality records that represent no true individual. Graph CG2 represents four individuals, one of whom is represented by a single instance.

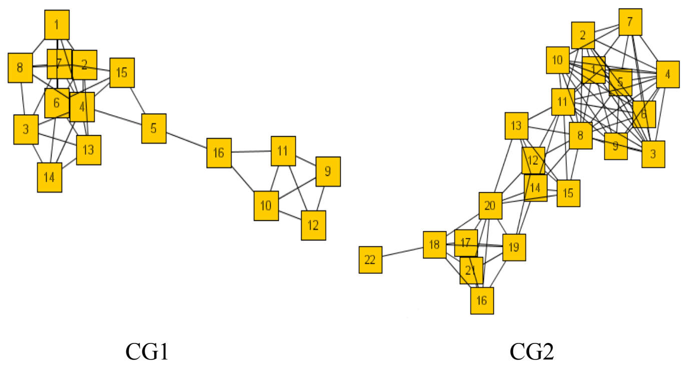

Figure 11. Customer Graphs CG1 and CG2.

The clustering results of SCAN, using the parameters (ε= 0.7, μ = 2), are presented in Figure 12. The results demonstrate that SCAN successfully found all the clusters and outliers.

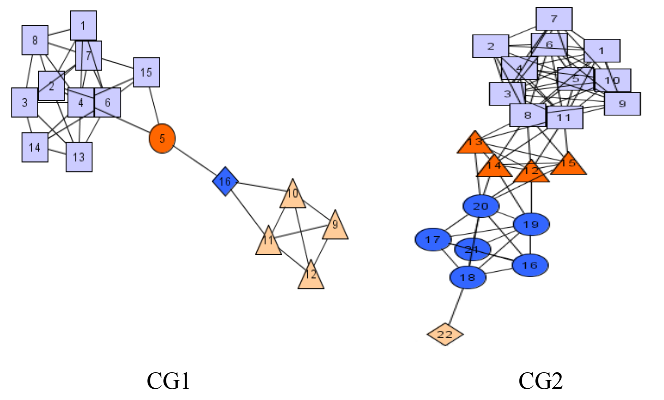

Figure 12. The Result of SCAN on CG1 and CG2.

The results of FastModularity are presented in Figure 13. It is clear that FastModularity failed to identify any outliers.

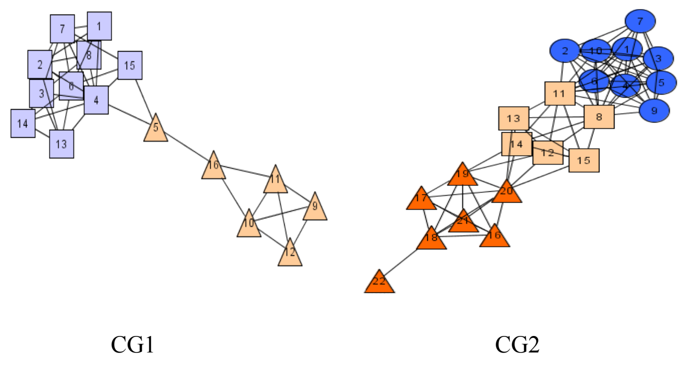

Figure 13. The Result of FastModularity on CG1 and CG2.

6.2.5 *Adjusted Rand Index Comparison*

As mentioned in Section 6.2.1, 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 1.

The ARI results clearly demonstrate that SCAN outperforms FastModularity at producing clustering that resemble the true clustering of the real world networks in our study.

Table 1. Adjust Rand Index Comparison.

	SCAN	**FastModularity**
College football	**1**	0.24
Political books	**0.71**	0.64
CG1	**1**	0.85
CG2	**1**	0.68

6.2.6 *Input Parameters*

SCAN algorithm uses two parameters: ε and μ. To choose them we adapted the heuristic suggested for DBSCAN in [10]. This involves making a k-nearest neighbor query for a sample of vertices and noting the nearest structural similarity as defined in Section 3.1. 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 graph 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 for μ, of 2.

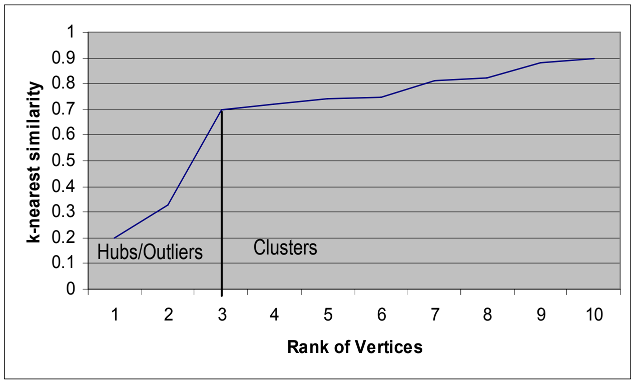

Figure 14. Sorted k-Nearest Structural Similarity.

7. CONCLUSIONS

Network clustering is a fundamental task in many fields of science and engineering. Many algorithms have been proposed from practitioners in different disciplines including computer science and physics. Successful examples are Min-Max Cut [4] and Normalized Cut [5], as well as Modularity-based algorithms [6][11][12]. 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 (outliers). Identifying hubs is essential for applications such as viral marketing and epidemiology. As vertices that bridge clusters, hubs are responsible for spreading ideas or disease. In contrast, outliers have little or no influence, and may be isolated as noise in the data.

In this paper, we proposed a method called SCAN (Structural Clustering Algorithm for Networks) to detect clusters, hubs and outliers in networks. SCAN clusters vertices based on their 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. 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* [13].

We applied SCAN 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 SCAN 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 algorithms.

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

8. REFERENCES

[1] S. Wasserman and K. Faust, "Social Network Analysis." Cambridge University Press, Cambridge (1994).

[2] R. Albert, H. Jeong, and A.-L. Barabási, "Diameter of the world-wide web." Nature 401, 130–131 (1999).

[3] J. M. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "The Web as a graph: Measurements, models and methods." In Proceedings of the International Conference on Combinatorics and Computing, number 1627 in Lecture Notes in Computer Science, pp. 1–18, Springer, Berlin (1999).

[4] 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.

[5] J. Shi and J. Malik, "Normalized cuts and image segmentation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 22, No. 8, 2000.

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

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

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

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

[10] 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.

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

[12] A. Clauset, M. E. J. Newman, and C. Moore, “Finding community in very large networks”, Physical Review E 70, 066111 (2004).

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

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

[15] L. Hubert and P. Arabie, “Comparing Partitions”. *Journal of Classification*, 193–218, 1985.

[16] 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.

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

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

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

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

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

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