A Novel Approach for Finding Clusters from Complex Networks

Dongming Chen
Software College of Northeastern University, Shenyang, China
Email: chendm@mail.neu.edu.cn

Xiaowei Xu
Department of Information Science in University of Arkansas at Little Rock, USA

***Abstract*—A feasible structural clustering method based on breadth-first-search is proposed for graphs. Clustering is very important and widely used in analyzing complex networks such as community identification. There are clusters with different shapes such as cliques and stars in practical application. Some existing algorithms can find clique-shaped clusters, but they are unable to identify star-shaped clusters that are familiar in scale free networks. A feasible solution is provided to solve the problem. It is superior to other algorithms in one or several of the following aspects: An algorithm without any input parameters, Running time on a network with *n* nodes and *m* links is *O*(*n*), Extracting clusters of mixed shapes.**

***Index Terms*—complex networks, clustering, modularity**

I. Introduction

Network clustering or graph partitioning is the division of a graph into a set of sub-graphs, called clusters. The term cluster analysis(first used by Tryon,1939) encompasses a number of different algorithms and methods for grouping objects of similar kind into respective categories. Clustering is very crucial to complex networks analysis [1-3]. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way". A cluster is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters. Clustering algorithms can be applied in many fields.

The paper is organized as follows. In section 2, we recall the related work for network clustering algorithms. We first formulize the conception of structure-connected clusters and then give the proposed clustering algorithm as well as its complexity analysis in section 3. Experimental results and evaluation of the algorithm are shown in section 4. Finally, we draw our conclusions and suggest future work in section 5.

II. Relation Work

A. Some Basic Concepts

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 as a priori. In this paper, we focus on simple, undirected, and unweighted graphs. 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. 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. After min-max cut, 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.

Modularity was proposed as a quality measure of network clustering [6,7]. A greedy method based on a hierarchical agglomeration clustering algorithm is proposed in [8]. Guimera and Amaral [9] optimize modularity using simulated annealing. But modularity also has resolution limit in community detection. Finding the maximal modularity is then equivalent to looking for the ideal tradeoff between the number of terms in the sum, i.e., the number of modules, and the value of each term [10]. D.M. Chen and X.W. Xu proposed an algorithm for

doi:10.4304/jsw.6.1.132-139

identifying useful structure in graphs clustering which is based on Breadth-First-Search[11].

B. A Breadth-First-Search Based Clustering Algorithm

The algorithm intends to identify clusters as well as hubs and outliers. Therefore, both connectivity and local structure are used in our definition of optimal clustering. Here we provide the concept of a structure-connected cluster, which extends a density-based cluster [3] and can distinguish good clusters, hubs and outliers in networks.

Let's consider social communities, people who share many friends form a community, and the more friends they have in common, the more intimate the community. Our method is based on common neighbors and not just direct connections. Two vertices are assigned to a cluster according to how they share neighbors. But in social networks there are different kinds of roles. There are people who are outsiders, and there are also people who are close to many communities but belong to none. The latter plays a special role in small-world networks as *hubs* [12] and it is illustrated by vertex *H* in Fig.1.

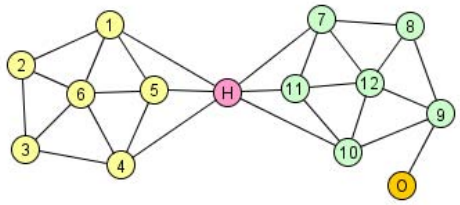

Figure 1. A small network with two clusters, a hub(vertex H) and an outlier(vertex O).

Here, we consider undirected and unweighted graphs. Given a graph $G=\{V,E\}$, where V is a set of vertices(or nodes); and E is set of pairs of distinct vertices, that is edges(or links). Before presenting the proposed method, some fundamental definitions employedd by the algorithm should be provided.

a. Problem Expression

The structure of a vertex is described by its neighborhood. A formal definition of vertex structure $\Gamma(v)$ is given as: Let $v \in V$, the structure of v is defined by its neighborhood, denoted by

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

Note that neighborhood of vertex v, $\Gamma(v)$, also includes v in addition to all neighbors of v. For instance, considering Fig.1, $\Gamma(1)$ would be {1, 2, 6, 5, H}. Having $\Gamma(v)$, now we can formulize similarity function, which is run for every edge, $\{v,w\} \in E$, in the network. We call the similarity function structural similarity because it is solely derived from vertex structure $\Gamma(v)$. The structural similarity between two vertices is measured by normalized common neighbors, which is also called cosine similarity commonly used in information retrieval. If we only use the number of shared neighbors, hub vertices, such as *H* in Fig.1, will be clustered into either of the clusters or two clusters will be unreasonably merged. Therefore, we normalize number of common neighbors by the geometric mean of the two neighborhoods' size. Note that In Fig.1, vertex *H* should be identified as a hub, shared in neighborhood of both clusters.

Utilizing vertex structure, we give structural Similarity $\sigma(v,w)$ which is denoted by:

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

When a member of a cluster shares a similar structure with one of its neighbors, their computed structural similarity will be large. Obviously structural similarity is symmetric, $\sigma(v,w)=\sigma(w,v)$. Structural similarity between v and w, $\sigma(v, w)$, would be greater than zero if and only if v and w are vertices of an edge $e \in E$. Under this circumstance, structural similarity takes values between 0 and 1. However, structural similarity should be restricted to control expansion of the cluster. Therefore, we apply a threshold ε to the computed structural similarity when assigning cluster membership, formulized in the following ε-neighborhood definition.

The ε-Neighborhood of a vertex $v \in V$ is denoted by: $N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$.

When a vertex shares structural similarity with enough neighbors, it becomes a *expander* or *seed* for a cluster. Such a vertex is called a kernel vertex. Kernel vertices are a special class of vertices that have a minimum of μ neighbors with a structural similarity that greater than or equals to the threshold ε. From kernel vertices we grow the clusters. In this way, only the parameters μ and ε determine the clustering of network. For a given ε, the minimal size of a cluster is determined by μ. If a vertex w is in ε-neighborhood of a kernel vertex v, vertex w should be included into the same cluster with vertex v, because they are connected and share a similar structure. This concept is known as direct structural reachability.

Direct structural reachability is symmetric for any pair of kernels. However, it is asymmetric if one of the vertices is not a kernel. Also the property of direct structural reachability is basis for the cluster expansion. A newly formed cluster C consists of a kernel vertex v and v's ε-neighborhood. Then we try to expand cluster C through any vertex w in v's ε-neighborhood. This approach guarantees that vertex w is directly structure-reachable from vertex v. Iterative queries for direct structural reachability usually adds more and more vertices into the current cluster.

A simple example is shown in Fig.2, in which similarities between vertices are given. Under the conditions of $\mu = 2$ and $\varepsilon = 0.6$, the possible scenario is as follows: We find *1* as the first kernel vertex since structural similarity between *1* and *2* is greater than ε, 0.6. Now a cluster of {*1*, *2*} is formed, and it should be expanded if possible. At the second step we look for any vertex that is similar to *2*. Among neighbors of *2*, vertex *5* is selected and inserted into the current cluster {*1*, *2*} due to similarity value of 0.75 between *2* and *5*. After the insertion, the cluster has now three vertices {*1*, *2*, *5*}. At

this stage of algorithm, it is noticeable that vertex *1* and *2* are kernel vertices; *2* is directly structure-reachable from *1*; and *5* is directly structure-reachable from *2*.

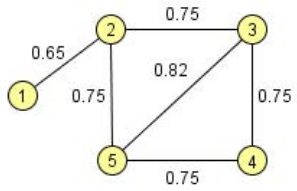

Figure 2. A simple network showing structural reachability.

After given example, we introduce another property of the proposed algorithm: structural reachability, which can be considered as chained form of direct structural reachability. The structural reachability is transitive, but it is asymmetric. It is only symmetric for a pair of kernels, as appears in previous example. More specifically, the structural reachability is a transitive closure of direct structural reachability.

Two non-kernel vertices in the same cluster may not be structure-reachable because the kernel condition may not hold for them. But they still belong to the same cluster because they both are structure-reachable from the same kernel. This idea is known as structural connectivity, and explained more formally as follows: 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. The structural 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. structural reachability.

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. The proposed algorithm finds all clusters w.r.t ε and μ. However, there might be some isolated vertices that are not assigned to clusters. If this is the case, we categorize each of those vertices either as a hub or an outlier. When an isolated vertex $v \in V$ has neighbors belonging to two or more different clusters, it is labeled as a hub vertex. Otherwise, an outlier.

In practice, the definitions of a hub and an outlier are flexible. It may be more useful to regard a hub 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. 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.

b. The Breadth-First-Search Based Algorithm

In this section, we describe the proposed algorithm which implements the search for clusters, hubs and outliers in complex network. The search begins by first visiting each vertex once to find structure-connected clusters.

Given graph $G=<V,E>$, parameters ε and μ which are described in the preceding section, the main process of the proposed algorithm is depicted as the following steps.

Step1: All vertices in V are marked as unclustered;

Step2: For each unclustered vertex $v \in V$, check whether v is a kernel vertex. If v's ε-neighborhood contains at least μ vertices, then v is a kernel vertex, or it is not a kernel vertex.

Step3: If v is a kernel, a new cluster with ClusterID is created, which expands from v.

Step 3.1: Starting with an arbitrary kernel and search for all vertices that are structure-reachable from v, thus the complete cluster containing vertex v is found. The new clusterID is assigned to all vertices found in this step.

Step 3.2: All vertices in ε-neighborhood of vertex v are inserted into a queue. For each vertex in the queue it computes all directly structural reachable vertices and inserts these vertices into the queue which are still unclustered.

Step 3.3: Repeat Step3.2 until the queue is empty.

Step4: If v is not a kernel, mark v as a non-member.

Step5: Go to Step2 until all vertices in V are checked.

Step6: Further classify each non-member vertex as hub or outlier. If a vertex has links to more than one cluster, it is be marked as a hub. Otherwise it is an outlier.

The algorithm performs one pass of a network and finds all structure-connected clusters for a given parameter setting (ε and μ). The non-member vertices can be further classified as hubs or outliers. This final classification is done according to what is appropriate for the network.

At the same time, the results of the proposed algorithm do not depend on the order of processed vertices, i.e. the obtained clustering of network is determinate, including number of clusters and association of kernels to clusters.

c. Experiment Results and Evaluation

We evaluate the proposed algorithm using both computer-generated network and real networks. The performance of the proposed algorithm is compared with FastModularity, a fast modularity-based network clustering algorithm proposed by Clauset *et al* in [8,13], which is faster than many competing algorithms: its running time on a graph with n vertices and m edges is $O(md\log n)$.

(1) Performance Analysis

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 [6].

We first empirically compared running time of the clustering algorithms. The experimental results demonstrated that the proposed algorithm is much faster with a linear running time in terms of the size of networks. The running time for FastModularity and the proposed algorithm on the synthetic graphs are plotted in Fig.3 and Fig.4. They show that the performance of the proposed

method is in fact linear w.r.t the number of vertices and the number of edges, while FastModularity's performance is basically quadratic and scales poorly for large graphs.

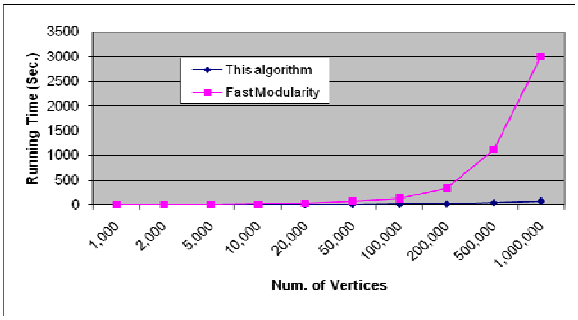

Figure 3. Running Time(Vertices).

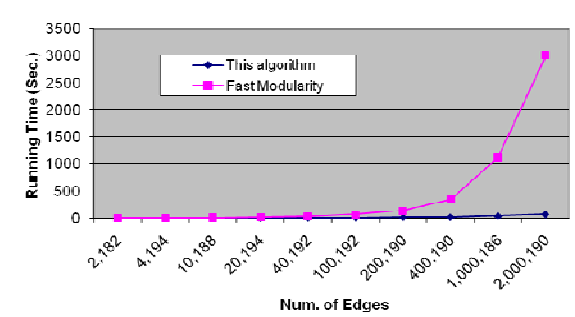

Figure 4. Running Time(Edges).

(2) Computer Generated Network

Fig.5 is a simple synthetic network with 17 vertices, including hubs and outliers. The clustering results of the proposed algorithm, using the parameters (ε=0.7, μ =2) are shown in Fig.6. The results demonstrate that the proposed algorithm successfully detected all the clusters(marked in different shapes: Parallelogram and Diamond) , hubs (node 1 and 3) and outliers(node 12,16 and 17).

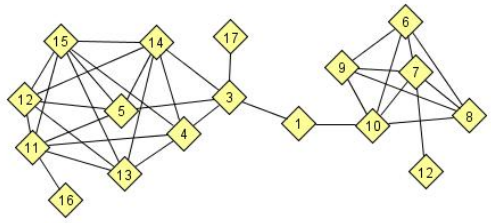

Figure 5. Synthetic network.

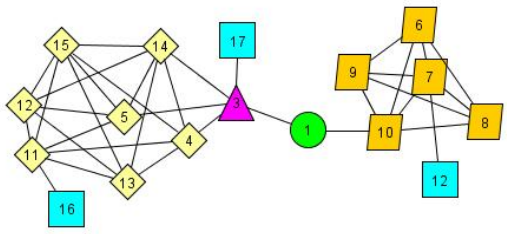

Figure 6. The result of the presented algorithm.

(3) Real network

① *Zachary Karate Club network*

The first real dataset is the Zachary Karate Club network. The karate club network is a social network consisting of 34 nodes representing people from a karate club at an American university and edges representing friendships between them. This network was compiled by Zachary [14] who was studying the social interactions between the members of the club. During the course of the study a dispute between the administrator of the club and the instructor of the club resulted in the split of the club into two. The instructor opened another club with about half the members from the original club. The original karate club network is shown in Fig.7, the round nodes indicate the instructors group and the square nodes indicate the administrators group. Fig.8 shows results clustered using the proposed algorithm. It exactly attains two clusters. More satisfying, hubs and outliers are correctly identified: Node colors correspond to different clusters; Triangle nodes(Node 3,9,14,32) denote Hubs; Diamond node(Node 12) denotes Outlier.

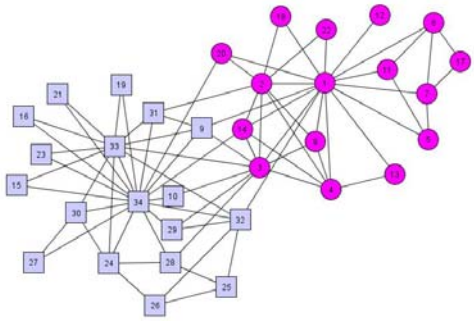

Figure 7. Original actual Zachary karate club dataset.

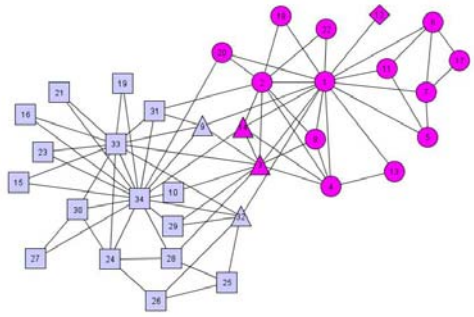

Figure 8. Clustering results of the Zachary karate club dataset.

② *Bottlenose Dolphins network*

The second example is the bottlenose dolphins network. The Community structure in the social network of bottlenose dolphins datasets contains an undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand, as assembled by Lusseau et al. in 2003 [15,16], extracted using the algorithm of Girvan and Newman [17]. In Fig.9, the squares and circles denote the primary split of the network into two groups. Traditional algorithm can detect clusters, but fail to denote hubs and outliers [17]. The proposed algorithm outperforms existing algorithm by detecting hubs(triangle nodes) and outliers(diamond nodes) as well as two clusters in accordance with the original, as shown in Fig.10.

Though the algorithm can find hubs and outliers and

outperforms other algorithms in speed, it needs two input parameters and is not suitable for dynamic network.

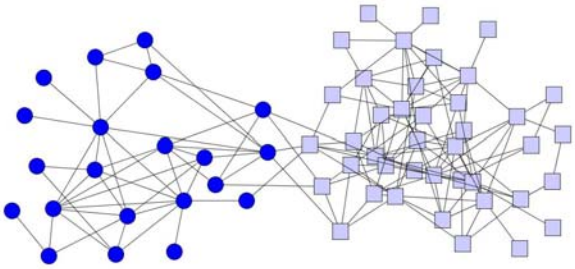

Figure 9. The Original Bottlenose Dolphins network.

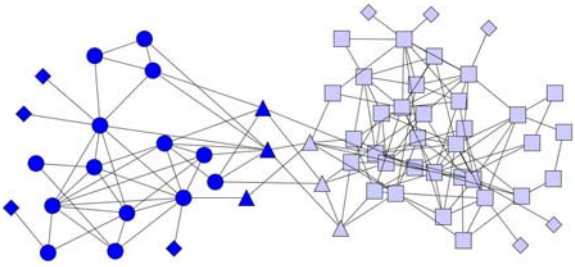

Figure 10. Bottlenose Dolphins network Clustered by the proposed algorithm.

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. The clusters are of various shapes such as cliques and stars. While all these network clustering methods successfully find clusters probably including clique-shaped clusters, they are generally unable to identify star-shaped clusters that are common in scale free networks. We propose a novel approach for finding clusters from complex networks in the followings sections.

III. The Novel Clustering Approach

A. Conception and Formulation

We present the novel clustering algorithm in this section. The main idea of our clustering algorithm is based on the observation that nodes within a cluster have more links connecting to nodes of the same cluster. This cliquishness of nodes in clusters is used to define our cluster. We assume that the network is represented as an undirected graph $G=<V, E>$, where V is set of nodes and E is set of undirected pairs of nodes.

Given a graph $G=<V,E>$, neighborhood of v is a set of nodes that are directly connected to v, formally:

$$N(v) = \{w|(v,w)\in E\} \qquad (1)$$

For any node $v\in V$, we measure the ability of v to attract neighbors. We call this ability "attractiveness", which is measured by the fraction of shared neighbors. Given a graph $G=<V,E>$, $v,w\in V$ and $w\in N(v)$, the *attractiveness* of v to w is measured by the fraction of neighbors of w and also neighbors of v, formally:

$$\alpha(v,w) = \frac{|N(v)\cap N(w)|+1}{|N(w)|} \qquad (2)$$

where $|\bullet|$ operator is the cardinality, i.e. the number of elements. The nominator is the cardinality of common neighbors plus v, because v is also a neighbor w.

The sufficient condition for w, ($w \in N(v)$) to be in the same cluster as v is that the number of links from $\{w, N(w)\}$ that connect to v must be more than half the total number of links radiating from w, formally: $\alpha(v, w) > 0.5$. In this case, w is called a *subordinate* of v. w is a *subordinate* of v, if w is a neighbor of v and $\alpha(v, w) > 0.5$. We call all subordinates of v the "subordinate neighborhood." The subordinate neighborhood of v is all the neighbors of v that are subordinates of v, formally:

$$N_\alpha(v) = \{w|w \in N(v) \wedge \alpha(v,w) > 0.5\} \qquad (3)$$

For any natural number k ($k\in\mathbb{N}^*$ and $\mathbb{N}^* = \{1, 2,\ldots\}$), k-subordinate neighbors of v are (k-1)-subordinates that satisfy the following condition:

$$N_\alpha^k(v) = \{w \mid w \in N_\alpha^{k-1}(v) \wedge \alpha^k(v,w) > 0.5\} \qquad (4)$$

where $\alpha^k(v,w) = \frac{|N_\alpha^{k-1}(v)\cap N(w)|+1}{|N(w)|}$ and $\alpha^1(v, w) = \alpha(v, w)$, $N_\alpha^1(v) = N_\alpha(v)$, and $N_\alpha^0(v) = N(v)$.

We need to identify nodes that can form a cluster with their subordinate neighbors. This ability, we call it "charisma," can be measured by the fraction of subordinate neighbors in the neighborhood. The 1-charisma of v is the fraction of subordinate neighbors in the neighborhood, formally:

$$\chi^1(v) = \frac{|N_\alpha(v)|}{|N(v)|} \qquad (5)$$

For any $k \in \mathbb{N}^*$, *k-charisma* of v is the fraction of k-subordinate neighbors in (k-1)-subordinate neighbors, formally:

$$\chi^k(v) = \frac{|N_\alpha^k(v)|}{|N_\alpha^{k-1}(v)|} \qquad (6)$$

where $\chi^1(v)= \chi(v)$. If $|N_\alpha^{k-1}(v)| = 0$, then $\chi^k(v) = 0$.

The maximal k-charisma is the charisma, which measures the ability of a node to build a cluster with its k-subordinate neighborhood. For any $k \in \mathbb{N}^*$, the maximal k-charisma is the charisma, formally:

$$\chi(v) = \max_k\{\chi^k(v)\} \qquad (7)$$

In graph theory, a path is a sequence of nodes such that from each of its nodes, there is a link to the next nodes in the sequence. Given a graph $G=<V, E>$, a path, p, is a sequence of nodes $p=<v_1, v_2, \ldots, v_n>$ such that $(v_i, v_{i+1})\in E$, where i =1, 2, …, n-1. A sub-graph is connected if there is a path between any pair of nodes in the component.

Given a graph $G=<V,E>$, a sub-graph $G'=<V',E'>$, where $V'\subseteq V$ and $E'\subseteq E$, is connected if $\forall(v,w)$, where v, $w \in V'$, $\exists s_i \in V'$, i=1,2,…k, such that $p = <v,s_1,s_2,\ldots,s_k,w>$ is a path(Connectedness).

A cliquishness set is a set of nodes where more than half of its links connects to nodes in the same set. Given a graph $G=<V,E>$, a set $C \subseteq V$ is cliquish if $\forall\ v\in C$, more than 50% of the neighbors of v, $N(v)$ are members of C, formally:

$$C = \{v \mid \forall v \in C : |C \cap N(v)| / |N(v)| > 0.5\} \quad (8)$$

A set of nodes, $C \subseteq V$, is a connected cliquish set if: (1) (Connectedness): there is a path between any pair of nodes in C; (2) (Cliquishness): for every node v in C, above 50% of $N(v)$ are members of C, formally:

$$\frac{|C \cap N(v)|}{|N(v)|} > 0.5 \quad (9)$$

For a given graph, a cluster is set of nodes that are connected, cliquish, as well as maximal in terms of connectedness and cliquishness.

A cluster is a connected cliquish set C, which is maximal (Maximality), i.e., there is no superset D, $D \supset C$, which satisfies both connectedness and cliquishness. We want to find nodes that can build a cluster with its k-subordinate neighbors. If a node can form a cluster with all its k-subordinate neighbors, we call it k-core. A formal definition of k-core is as follows: For any k and $i \in \mathbb{N}^*$, a node v is a k-core if it satisfies both of the following conditions: (1) $k = \arg\min_i \left(\chi^i(v) = 1\right)$; (2) $\frac{|N_\alpha^k(v)|}{|N(v)|} > 0.5$. We call k the core-level of v. The first condition above implies all subordinate neighbors of v being cliquish. The second condition implies over 50% of v's neighbors are cliquish, too. Therefore, v can build a cluster with its subordinate neighbors.

There may be nodes not belonging to any clusters due to the violation of either connectedness or cliquishness of a cluster definition. We call them "nonmembers" because they are not members of any cluster, and they are either isolated (called outliers) or connect multiple clusters with cliquishness to any of them less than or equal to 50%(called hubs).

B. The Novel Algorithm

In this section, we describe the proposed algorithm, which implements the search for clusters in a network or graphs. The clusters have to satisfy connectedness, cliquishness, and maximality conditions based on our cluster definition. The general process of the proposed Algorithm(named NovelCluster) includes four steps:

Step 1: All nodes in V are marked as unclassified.

Step 2: Calculate core levels for all nodes. For all $v \in V$, test if v is a k-core in terms of $\chi^k(v) \neq 1 \vee |N_\alpha^k(v)| / |N(v)| \leq 0.5$. If the condition satisfies, it means that v is a k-core, then insert v into an array core[k]. The k-cores of the same core-level k are inserted into the same array.

Step 3: The k-cores of the same core-level k are inserted into the same array, so finding clusters starting from k-cores with the smallest core level k. For all v in core[i], if v is unclassified, create a new cluster C and assign it a new Cluster ID. Let $C = \{v\} \cup N_\alpha^k(v)$ (here, $N_\alpha^k(v)$ hasn't been clustered), and label all nodes in C as classified, then goto next step (expand initial cluster C by its neighbors).

Step4: Maximally expanding C with its neighbors, it is a breadth-first-expansion of the current cluster. (1) For all neighbors of C, if v is unclassified, insert candidate v into queue Q; (2) Expand C with candidate in Q. If Q is not empty, take the following two steps repeatedly: (1) Remove first element of Q; (2) Check for cliquishness. If $|C \cap N(v)| / |N(v)| > 0.5$, insert v into C. For all $w \in N(v)$, if w is unclassified, then w is a new candidate for C.

The clustering procedure repeated for all core-levels in ascending order until all k-cores are examined. After the clustering procedure is finished, the unclassified nodes can be further classified as either outliers or hubs. Some application does not allow outliers or hubs. In this case, we can assign all nonmembers to clusters based on the number of links to the clusters weighted by the charisma. The core's charisma is 1 based on our definition.

From above, we can draw the following three conclusions: (1) The discovered clusters by the algorithm satisfy all conditions in our cluster definition: connectedness, cliquishness, and maximality; (2) The results of the algorithm do not depend on the order of processed vertices; (3) It is clear that the proposed algorithm doesn't require any input parameter, which is advantageous for dynamic networks.

C. Complexity Analysis

Given a graph with m edges and n vertices, the proposed method first finds all structure-connected clusters w.r.t. a given parameter setting by checking each vertex of the graph. 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 $\deg(v_i)$ is the degree of vertex v_i, $i = 1,2,\ldots,n$. 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. In the worst case the graph is complete and the running time is $O(n(n\text{-}1))$, or $O(n^2)$. However, real networks are generally sparse with a power law degree distribution [18]. The expected number of neighobors is a constant [19]. For these networks the running time will be in liner proportion to the number of vertices in the graph, i.e. $O(|V|)$.

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:

$$p(k) = \binom{n}{k} p^k (1-p)^{n-k} \approx \frac{z^k e^k}{k!} \quad (10)$$

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 proposed algorithm is $O(n)$. Therefore, the complexity

in terms of the number of edges in the graph for the proposed algorithm is in general linear. The complexity in terms of the number of vertices is quadratic in the worst case of a complete graph.

IV. Experimental Results and Evaluation

In this section we evaluate the algorithm using both synthetic and real datasets. The performance of the proposed algorithm is compared with FastModularity, a fast modularity-based network clustering algorithm proposed by Clauset *et al* in [8], 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 the new algorithm in C++. We used the original source code of FastModularity by Clauset *et al* [13].

The 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 [6], 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. Figure 11 shows this network with schools in the same conference identified by color using the proposed method. Figure 12 is the similar result using the FastModularity Algorithm.

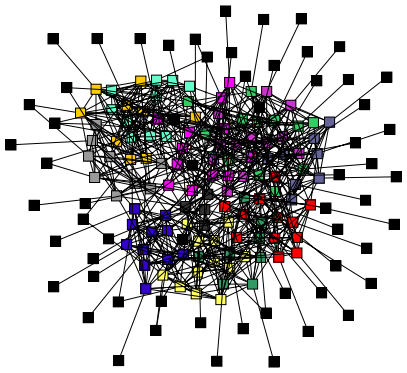

Figure 11. NCAA Football Bowl Subdivision Schedule Clustered by the Proposed Algorithm

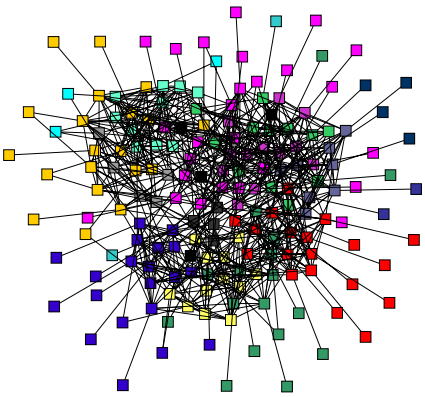

Figure 12. NCAA Football Bowl Subdivision Schedule Clustered by FastModularity Algorithm

From Figure 11 and Figure 12, we can see the proposed method can detect outliers, but the FastModularity algorithm just classify these outliers into different clusters.

To evaluate the ability to detect clusters of various shapes, we generate synthetic networks, consisting of both cliques and stars. One example of the networks is shown in Figure 13 to Figure 15. The clustering result of the proposed method (Figure 13), SCAN [19](Figure 14)), and CMN [4] (Figure 15) is plotted using colors to represent clusters. The results demonstrate that only the proposed method can accurately detect both star- and clique-shaped clusters simultaneously.

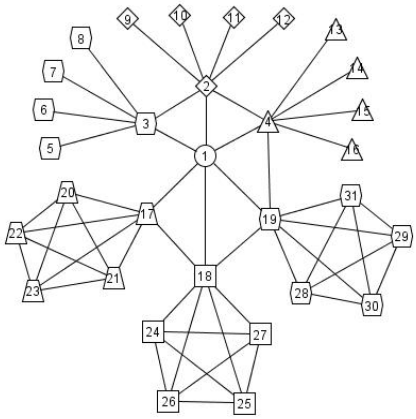

Figure 13. The clustering results of the proposed algorithm

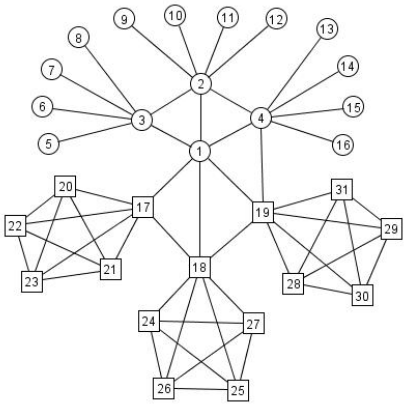

Figure 14. The clustering results of SCAN

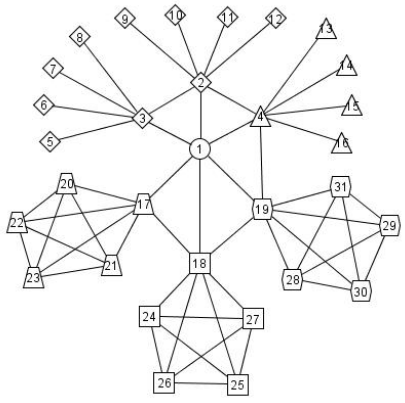

Figure 15. The clustering results of CMN

V. Conclusions and Future Work

In this paper, we propose a novel structural clustering method for graphs/networks. Through theoretical analysis and extensive experiments, we can conclude that the proposed method is significantly more accurate and efficient than other algorithms, especially in detecting the star-shaped clusters that are common in scale free networks.

Data clustering is a field of active research in machine learning and data mining. Most of the work has focused on static data sets. There has been little work on clustering of dynamic data. In the future, we will study clustering algorithm on dynamic data set as a set of elements whose parameters change over time. A flock of flying birds is an example of a dynamic data set. We will be interested in exploring algorithms are capable of finding relationships among the elements in a dynamic data set.

Acknowledgment

The authors wish to thank Zachary and Lusseau et al. This work was supported in part by the National Natural Science Foundation of China under Grant No.60872040, and the Natural Science Foundation of Liaoning Province of China under Grant No.20082037

References

[1] Zach Solan, David Horn, Eytan Ruppin, Shimon Edelman, “Unsupervised learning of natural languages”, PNAS, vol. 102, no.33, pp. 11629–11634, 2005.
[2] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu, “Incremental Clustering for Mining in a Data Warehouse Environment”, In Proceeding of 24th VLDB Conference, 1998.
[3] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, In Proceeding Of KDD, pp.226-231, 1996.
[4] C. Ding, X. He, H. Zha, M. Gu, H. Simon, “A min-max cut algorithm for graph partitioning and data clustering”, In Proceeding of ICDM 2001.
[5] J. Shi, J. Malik, “Normalized cuts and image segmentation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, 2000.
[6] M. E. J. Newman, M. Girvan, “Finding and evaluating community structure in networks”, Phys. Rev. E 69, 026113, 2004.
[7] M. E. J. Newman, “Modularity and community structure in networks”, PNAS, vol. 103, no. 23, pp. 8578-8582, 2006.
[8] A. Clauset, M. E. J. Newman, C. Moore, “Finding community structure in very large networks”, Physical Review E 70, 066111, 2004.
[9] R. Guimera, L. A. N. Amaral, “Functional cartography of complex metabolic networks”, Nature, vol. 433, pp.895–900, 2005.
[10] Santo Fortunato, Marc Barthelemy, “Resolution limit in community detection”, PNAS, vol. 104, no.1, pp.36-41, 2007.
[11] D.M. Chen and X.W. Xu, “An algorithm for identifying useful structure in graphs clustering”, 2010 Second International Workshop on Education Technology and Computer Science, vol. 1, pp. 63-66, Wuhan, China, 2010.
[12] Watts DJ, Strogatz SH. Collective dynamics of 'small-world' networks. *Nature*, 1998, 393:440-442.
[13] http://cs.unm.edu/~aaron/research/fastmodularity.htm.
[14] W. W. Zachary. An information flow model for conflict and fission in small groups. Journal of Anthropological Research, 1977, 33: 452-473.
[15] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, S. M. Dawson. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations-Can geographic isolation explain this unique trait? Behavioral Ecology and Sociobiology, 2003, 54(4):396-405.
[16] D. Lusseau. The emergent properties of a dolphin social network. In Proc. R. Soc. London B, 2003 (sup.), 270:186-188.
[17] M. Girvan, M. E. J. Newman. Community structure in social and biological networks. PNAS, 2002, 99(12):7821–7826.
[18] A.L. Barabasi, R. Albert, “Emergence of scaling in random networks”, Science, vol. 286, pp.509-512, 1999.
[19] X. Xu, N. Yuruk, Z. Feng, T. A. J. Schweiger, “SCAN: A Structural Clustering Algorithm for Networks”, In Proceeding of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, CA, 2007.
[20] L. Hubert, P. Arabie, “Comparing Partitions. Journal of Classification”, vol. 2, pp.193–218, 1985.

Dongming Chen is an associate professor in software college of Northeastern University, China. He was awarded the Ph.D. degree in 2006. His research areas include complex networks, information mining, information security. He has undertaken the National Natural Science Foundation of China, the Natural Science foundation of Liaoning Province and other projects. He is Editors Committee Member of the International Journal of Advancements in Computing Technology, International Committee member of AICIT's International Conference Series, Program committee of the 2010 International Conference on Computer Application and System Modeling. He is a member of CCF.

XiaoweiXu is a professor in Department of Information Science in University of Arkansas at Little Rock, USA. He was awarded the Ph.D. degree at University of Munich, Germany in 1998. His research interests cover Data Mining, Machine Learning and Knowledge Discovery. He has undertaken the American NIH/NCRR through Department of Health and Human Services(DHHS), FDA, German Federal Ministry of Education and Research(BMBF), and other projects. He is Program Committee Member of many international conferences and Referee for Journal and Conference Submissions. He is a member of ACM, AIS, SIM.