gSkeletonClu: Density-based Network Clustering via Structure-Connected Tree Division or Agglomeration

Heli Sun*, Jianbin Huang†, Jiawei Han‡, Hongbo Deng‡, Peixiang Zhao‡ and Boqin Feng*
*Department of Computer Science, Xi'an Jiaotong University, Xi'an, China
Email: helisun@stu.xjtu.edu.cn, bqfeng@mail.xjtu.edu.cn
†School of Software, Xidian University, Xi'an, China
Email: jbhuang@xidian.edu.cn
‡Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
Email: hanj@cs.uiuc.edu, hbdeng@uiuc.edu, pzhao4@illinois.edu

***Abstract*—Community detection is an important task for mining the structure and function of complex networks. Many pervious approaches are difficult to detect communities with arbitrary size and shape, and are unable to identify hubs and outliers. A recently proposed network clustering algorithm, SCAN, is effective and can overcome this difficulty. However, it depends on a sensitive parameter: minimum similarity threshold ε, but provides no automated way to find it. In this paper, we propose a novel density-based network clustering algorithm, called gSkeletonClu (graph-skeleton based clustering). By projecting a network to its Core-Connected Maximal Spanning Tree (CCMST), the network clustering problem is converted to finding core-connected components in the CCMST. We discover that all possible values of the parameter ε lie in the edge weights of the corresponding CCMST. By means of tree divisive or agglomerative clustering, our algorithm can find the optimal parameter ε and detect communities, hubs and outliers in large-scale undirected networks automatically without any user interaction. Extensive experiments on both real-world and synthetic networks demonstrate the superior performance of gSkeletonClu over the baseline methods.**

***Keywords*-Density-based Network Clustering; Community Discovery; Hubs and Outliers; Parameter Selection**

I. Introduction

Nowadays, many real-world networks possess intrinsic community structure, such as large social networks, Web graphs, and biological networks. A community (also referred to as module or cluster) is typically thought of as a group of nodes with dense connections within groups and sparse connections between groups as well. Community discovery within networks is an important problem with many applications in a number of disciplines ranging from social network analysis to image segmentation and from analyzing protein interaction networks to the circuit layout problem.

Finding communities in complex networks is a nontrivial task, since the number of communities in the network is typically unknown and the communities often have arbitrary size and shape. Moreover, besides the cluster nodes densely connected with communities, there are some nodes in special roles like hubs and outliers. As we know, hubs play important roles in many real-world networks. For example, hubs in the WWW could be utilized to improve the search engine rankings for relevant authoritative Web pages [1], and hubs in viral marketing and epidemiology could be central nodes for spreading ideas or diseases. On the contrary, outliers are marginally connected with the community members. Since the characteristics of outliers deviate significantly from the communities, they should be isolated as noise. Therefore, how to detect communities as well as hubs and outliers in a network becomes an interesting and challenging problem.

Most existing approaches only study the problem of community detection without considering hubs and outliers. A density-based network clustering algorithm SCAN [2] can overcome this difficulty. However, it needs user to specify a minimum similarity ε and a minimum cluster size μ to define clusters, and is sensitive to the parameter ε which is difficult to determine. Actually, how to locate the optimal parameter ε automatically for the density-based clustering methods (e.g., DBSCAN [3] and SCAN) is a long-standing and challenging task.

The connectivity structure of a large-scale network is highly complex, which makes the selection of optimal parameter ε difficult. However, we have found that a cluster defined by density-based clustering algorithms is composed of two types of objects: cores and borders. The cluster can be determined by the density-connected cores embedded in it uniquely. Hence, once all the components of connected cores have been detected, all clusters can be revealed. Accordingly, we convert the problem of detecting ε-clusters in a network to finding core-connected components in the network weighted by a new measurement: core-connectivity-similarity. It is equal to partitioning the core-connected network with a minimal threshold ε (i.e., remove all the edges whose weights are below ε from the network).

Actually, the problem above can be easily solved by the Maximal Spanning Tree (MST), a connectivity skeleton of the network [4]. To motivate this, we illustrate a schematic network in Figure 1(a). Given $\varepsilon = 3$, if all the edges with weights no more than current ε are removed from the network, two unconnected sub-graphs will emerge, as shown in Figure 1(c). In contrast, if we first construct the MST of the network, as shown in Figure 1(b), and then partition the MST with the same threshold, the partitioning result on the MST is equal to that on the original network. In the

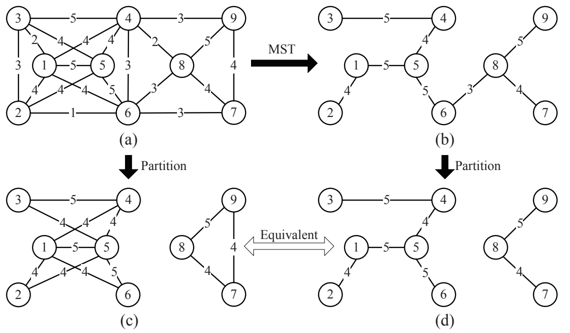


Figure 1. A schematic network: (a) the original network, (b) the MST of the network, (c) the partition of the original network with $\varepsilon = 3$, (d) the partition of the MST with $\varepsilon = 3$.

same way, the core-connected components can be detected on the Core-Connected Maximal Spanning Tree (CCMST) of the network. Since each edge of the CCMST is a cut edge, different edge weights will lead to different partition results and form different clusters. It also means that all possible ε, which we can adopt to cluster the network, can be found in the edge weights of the CCMST. Furthermore, the best clustering result can be selected by a quality function.

In this paper, a novel density-based network clustering algorithm, called gSkeletonClu, is proposed to perform the clustering on the CCMST. To our best knowledge, our work is the first that builds the connection between the density-based clustering principle and the MST-based clustering method. The main contributions of this paper are summarized in the following:

1) By projecting the network to its CCMST, we convert the community discovery problem to finding core-connected components in the CCMST. Consequently, we only need to deal with the essential edges of the network which reduces the size of the candidate solution space significantly. We have proven that the clustering result on the CCMST is equal to that on the original network.
2) We discover that all possible values of the parameter ε lie in the edge weights of the corresponding CCMST. Using the modularity measure Q_s, our algorithm can select the optimal parameter ε automatically, without any user interaction.
3) Our algorithm not only uncovers the communities within the network, but also identifies the hubs and outliers. Experimental results show that our algorithm is scalable and achieves high accuracy.
4) Our algorithm can overcome the chaining effect that the traditional MST clustering methods suffer from and the resolution limit possessed by other modularity-based algorithms.

The rest of the paper is organized as follows. First we briefly review the related work in Section 2. In section 3, we introduce the concepts of structural-connected clusters. In section 4, we formulize the notion of clusters derived from the CCMST and present the theoretical analysis. In section 5, we describe the algorithms in detail. In section 6, we report the experimental results. Finally, we summarize our conclusions and suggest future work in section 7.

II. Related work

The problem of detecting communities within networks has been extensively studied for years. Graph partitioning is a natural choice for this problem, such as Kernighan-Lin algorithm [5], Metis [6], and normalized cut [7]. To make the calculation of cut functions more efficient, spectral clustering method has been proposed where the eigenvectors of certain normalized similarity matrices are used for the clustering purpose [8]. Since the community structure in networks is highly complex, new clustering methods have recently been introduced to solve this challenging problem.

Density-based Clustering Methods: Density-based clustering approaches (e.g., DBSCAN [3] and OPTICS [9]) have been widely used in data mining owing to their ability of finding clusters of arbitrary shape even in the presence of noise. Recently, Xu *et al.* proposed a structural network clustering algorithm SCAN [2] extended from DBSCAN. This algorithm can find communities as well as hubs and outliers. However, the main difficulty for the SCAN algorithm is that it is sensitive to the minimal threshold ε of the structure-similarity. A simple "knee hypothesis" has been presented to locate the parameter ε manually for the SCAN algorithm. However, the knee dose not always correspond to the optimal ε value. More importantly, there are no obvious knees on the *k*-nearest ranked similarity plot for most of the real-world networks. To deal with this problem, Bortner *et al.* proposed a new algorithm, called SCOT+HintClus [10], to detect the hierarchical cluster boundaries of network through extension of the algorithm OPTICS [9]. However, it does not find the global optimal ε. Our work tries to solve the sensitive parameter problem of density-based network clustering from another angle.

Graph-theoretical Clustering Methods: Clustering algorithms based on graph theory can be used to detect clusters of different shapes and sizes. One of the best-known graph-based hierarchical clustering algorithms is based on the construction of the minimal (or maximal) spanning tree (MST) of the objects which was initially proposed by Zahn [11]. The standard MST divisive algorithm removes edges from the MST in order of decreasing length until the specified number of clusters results. A clustering algorithm using an MST takes the advantage that it is able to only consider the necessary connections between the data patterns and the cost of clustering can be decreased. However, the number of the desired clusters should be given in advance. Moreover, simple linkage-based methods often suffer from the problem of chaining effect. Some clustering algorithms have been shown closely related to MST, such as Single-link [12]. Recently, some researchers utilized the MST-based clustering method to analyze complex networks [13].

Our work introduces tree clustering into the framework of density-based clustering which is rather different from the traditional MST-based clustering method.

Quality Functions for Network Clustering: Clustering validity checking is an important issue in cluster analysis [14]. For community detection, the most popular quality function is the modularity measure Q proposed by Newman and Girvan [15]. Recently, a similarity-based modularity function Q_s was presented by Feng *et al.* [16] through extension from the connection-based modularity Q, which has a better ability to deal with hubs and outliers. The Q_s function is defined as follows:

$$Q_s = \sum_{i=1}^{k} \left[\frac{IS_i}{TS} - \left(\frac{DS_i}{TS} \right)^2 \right],$$

where k is the number of clusters, $IS_i = \sum_{u,v \in C_i} \sigma(u,v)$ is the total similarity of nodes within cluster C_i, $DS_i = \sum_{u \in C_i, v \in V} \sigma(u,v)$ is the total similarity between nodes in cluster C_i and any node in the network, and $TS = \sum_{u,v \in V} \sigma(u,v)$ is the total similarity between any two nodes in the network.

Modularity optimization itself is a popular method for community detection. Most modularity-based algorithms find the optimal clustering result via greedily maximizing the value of Q, such as FastModularity [17] and Simulated Annealing (SA for short) [18]. However, recent research shows that modularity is not a scale-invariant measure, and hence, by relying on its maximization, detection of communities smaller than a certain size is impossible. This serious problem is well known as the resolution limit [19]. Compared with the traditional modularity-based methods, our work uses modularity as a quality function to guide the selection of optimal clustering results.

III. Preliminaries

Given a minimum similarity ε and a minimum cluster size μ, the main idea of structure-connected clustering is that for each node in a cluster, it must have at least μ neighbors whose structural similarities are at least ε. In this section, we formulize the notion of a structure-connected cluster, which extends that of a density-based cluster [3], [9].

Definition 1. *(Structural Similarity) Let $G = (V, E, w)$ be a weighted undirected network and $w(e)$ be the weight of the edge e. For a node $u \in V$, we define $w(\{u, u\}) = 1$. The structure neighborhood of a node u is the set $\Gamma(u)$ containing u and its adjacent nodes which are incident a common edge with $u : \Gamma(u) = \{v \in V | \{u, v\} \in E\} \cup \{u\}$. The* structural similarity *between two adjacent nodes u and v is then*

$$\sigma(u,v) = \frac{\sum\limits_{x \in \Gamma(u) \cap \Gamma(v)} w(u,x) \cdot w(v,x)}{\sqrt{\sum\limits_{x \in \Gamma(u)} w^2(u,x)} \cdot \sqrt{\sum\limits_{x \in \Gamma(v)} w^2(v,x)}}. \quad (1)$$

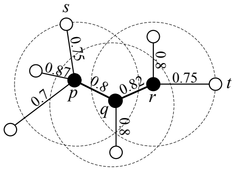


Figure 2. The structure reachablility and structure connectivity relationship of the nodes in a density-based cluster.

The above structural similarity is extended from a cosine similarity used in [2] which denotes the local connectivity density of any two adjacent nodes in an undirected network. It can be replaced by other similarity definitions such as Jaccard similarity, and our experimental results show that the cosine similarity is better.

Definition 2. *(Core) For a node u, the set of neighbors having structural similarity greater than ε forms the ε-neighborhood of u:*

$$\Gamma_\varepsilon(u) = \{v \in \Gamma(u) | \sigma(u,v) \geq \varepsilon\}. \quad (2)$$

If $|\Gamma_\varepsilon(u)| \geq \mu$, then u is a core node*, denoted by $\mathrm{K}_{\varepsilon,\mu}(u)$.*

A core node is the one whose ε-neighborhood is at least μ. As shown in Figure 2, when we set $\varepsilon = 0.75$ and $\mu = 4$, node p is a core and node q is in the ε-neighborhood of p. According to the principle of density-based clustering, the clusters grow from core nodes. If a node is in the ε-neighborhood of a core, it should be in the same cluster with the core. This idea is formulized in the following definition of direct structure reachability.

Definition 3. *(Structure-Reachable) Given $\varepsilon \in \mathbb{R}$, $\mu \in \mathbb{N}$, a node $v \in V$ is* directly structure-reachable *from a node $u \in V$ iff $\mathrm{K}_{\varepsilon,\mu}(u) \wedge v \in \Gamma_\varepsilon(u)$, denoted by $u \mapsto_{\varepsilon,\mu} v$. A node $v \in V$ is* structure-reachable *from $u \in V$ iff $\exists\{u_1, ..., u_n\} \subseteq V$* s.t. *$u = u_1, v = u_n$, and $\forall i \in \{1, 2, ..., n-1\}$ such that $u_i \mapsto_{\varepsilon,\mu} u_{i+1}$. This is denoted by $u \rightarrow_{\varepsilon,\mu} v$.*

A node $v \in V$ is directly structure-reachable from a core node $u \in V$ if $v \in \Gamma_\varepsilon(u)$. Obviously, directly structure-reachable is symmetric for pairs of core nodes. If a non-core node v is directly structure-reachable from a core node u, then v is a *border node* attached to u. In general, directly structure-reachable is not symmetric if a core node and a border node are involved.

We depict the notion of structure-reachable in Figure 2. When we set $\varepsilon = 0.75$ and $\mu = 4$, there are four nodes in the neighborhood of node p, q and r respectively, whose structure similarities with them are greater than current ε, thus p, q and r are all core nodes. Since the structure similarity between p and q is greater than 0.75, p and q are directly structure-reachable from each other. And for the same reason, r and q are also directly structure-reachable from each other. According to Definition 3, core nodes p, q and r are structure-reachable from each other. However, the border node s is only structure-reachable from the three core nodes on one side.

The transitive closure of directly structure-reachable relation forms clusters, and any pair of nodes in the same cluster is *structure connected*.

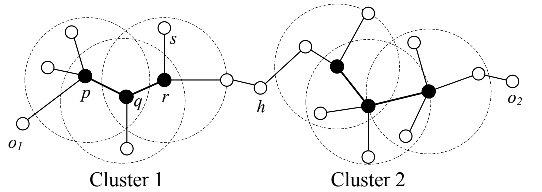


Figure 3. An example network with two structure-connected clusters, one hub and two outliers.

Definition 4. *(Structure-Connected Cluster) The set $C[u] \subseteq V$ is a cluster represented by $\mathrm{K}_{\varepsilon,\mu}(u) \in V$ iff (1)$u \in C[u]$; (2)$\forall v \in V$, $u \rightarrow_{\varepsilon,\mu} v \Rightarrow v \in C[u]$; and (3)$|C[u]| \geq \mu$.*

A cluster C contains exactly the nodes which are structure-reachable from an arbitrary core in C. If there are two core nodes $u, v \in V$ *s.t.* $u \rightarrow_{\varepsilon,\mu} v$, then $C[u] = C[v]$. Consequently, a cluster is uniquely determined by the connected core nodes in it. As shown in Figure 3, cluster 1 contains all the nodes which are structure-reachable from core nodes p, q and r. Thus, $C[p] = C[q] = C[r]$. There also may exist some border nodes (e.g., node s in Figure 3).

Given parameters ε and μ, a *clustering* of a network G is the set $CR_{\varepsilon,\mu}$ of distinct structure-connected clusters found in the network. After the network is clustered, there are still some nodes that are not suitable to be assigned to any cluster and they may be hubs or outliers.

Definition 5. *(Hub and Outlier) Given parameters ε, μ, and a clustering $CR_{\varepsilon,\mu}$ of the network G, a node $h \in V$ is a* hub *iff (1) h does not belong to any cluster: $\forall C[u] \in CR_{\varepsilon,\mu}$, $h \notin C[u]$; (2) h bridges multiple clusters: $\exists C, D \in CR_{\varepsilon,\mu}$, $C \neq D$, $u \in C \wedge v \in D$, s.t. $h \in \Gamma(u) \wedge h \in \Gamma(v)$. Any node that is not in a cluster and not a hub is called an* outlier.

As shown in Figure 3, node h that connects two nodes of different clusters is regarded as a hub, and nodes o_1 and o_2, which are connected with only one cluster, should be regarded as outliers.

IV. Clusters Derived from Core-Connected MST

To introduce the notion of core-derived clusters, we make the following observation: each cluster is determined by the structure-connected core nodes in it. Given $\{u, v\} \in E$, we must decide whether u and v are core nodes w.r.t. current ε and whether u and v are structure-reachable from each other. If so, u and v will lie in the same cluster. Our new algorithm gSkeletonClu bases on the principle of finding the core-connected components at different ε levels. The involved concepts are introduced in the following.

A. Core Connectivity Similarity

Definition 6. *(Core-Similarity) Given a node $u \in V$, the* core-similarity *of u is*

$$CS(u) \equiv \begin{cases} \max\{\varepsilon \in \mathbb{R}^+ : \\ |\{v \in \Gamma(u) : \sigma(u,v) \geq \varepsilon\}| \geq \mu\} & |\Gamma(u)| \geq \mu \\ 0 & otherwise. \end{cases} \quad (3)$$

The core-similarity of a node u is the maximum of structural-similarity $\tilde{\varepsilon}$ such that u would be a core node w.r.t. $|\Gamma_{\tilde{\varepsilon}}(u)| \geq \mu$. Otherwise, the core-similarity is zero. As shown in Figure 4(a), the core-similarity of node p is 0.75 when $\mu = 4$, because the size of the ε-Neighborhood of node p is just four when $\varepsilon = 0.75$ and node p will no longer be a core when $\varepsilon > 0.75$.

Definition 7. *(Reachability-Similarity) Given u, $v \in V$, the* reachability-similarity *of v w.r.t. u is*

$$RS(u,v) \equiv \min\{CS(u), \sigma(u,v)\}. \quad (4)$$

Intuitively, the reachability-similarity of a node v w.r.t. a core node u is the maximum of structural similarities such that v is directly structure-reachable from u. As shown in Figure 4(a), the reachability-similarity of node q w.r.t. core node p is 0.75 which is equal to the core-similarity of node p, because the similarity between the two nodes is not less than the core-similarity of p. While the reachability-similarity of node r w.r.t. core node p is 0.7 which is equal to the similarity between p and r.

Definition 8. *(Core-Connected) Given $\varepsilon \in \mathbb{R}$, $\mu \in \mathbb{N}$, $u, v \in V$, u and v are* directly core-connected *with each other iff $\mathrm{K}_{\varepsilon,\mu}(u) \wedge \mathrm{K}_{\varepsilon,\mu}(v) \wedge u \mapsto_{\varepsilon,\mu} v$. This is denoted by $u \leftrightarrow_{\varepsilon,\mu} v$.*

Since directly structure-reachable is symmetric for any pair of core nodes, if two core nodes are directly structure-reachable from each other, then they are *directly core-connected*. The transitive closure of the directly core-connected relation forms core-connected components, and any pair of nodes in the same component are *core-connected*.

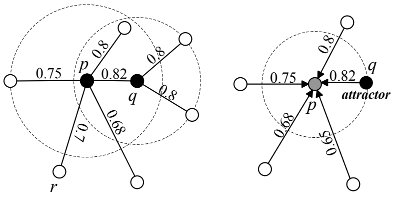
Definition 9. *(Core-Connectivity-Similarity) Given $\{u, v\} \in E$, the* core-connectivity-similarity *of u and v is*

$$\begin{aligned} CCS(u,v) &\equiv \min\{RS(u,v), RS(v,u)\} \\ &\equiv \min\{CS(u), CS(v), \sigma(u,v)\} \end{aligned} . \quad (5)$$

The core-connectivity-similarity of two nodes is the minimum of their core-similarities and the structure-similarity between them. As shown in Figure 4(a), nodes p and q are core-connected when $\varepsilon = 0.75$ and $\mu = 4$, because p and q are both core nodes and their structure-similarity is not less than the current ε. But if we set current $\varepsilon > 0.75$, nodes p and q are not core-connected any more, because p will not be a core under this configuration. Thus, the core-connectivity-similarity of p and q is 0.75 when $\mu = 4$. The following Theorem 1 shows that the core-connectivity-similarity of two nodes is the maximal structural similarity such that they are both core nodes and directly core-connected from each other.

Theorem 1. *Given a network $G = (V, E, w)$, $\{u, v\} \in E$, $\mu \in \mathbb{N}$, if $CCS(u,v) = \hat{\varepsilon}$, then $\hat{\varepsilon}$ is the maximum of ε s.t. $u \leftrightarrow_{\varepsilon,\mu} v$.*

For each edge $e = \{u, v\} \in E$ in a weighted network $G = (V, E, w)$, we calculate the core-connectivity-similarity for each pair of adjacent nodes u and v. Set $\varpi(e) = CCS(u,v)$. Then we get a core-connected network



(a) (b)

Figure 4. (a)an example for calculating core-similarity and reachability-similarity; (b)an example for calculating attachability-similarity and finding attractor.

$G = (V, E, \varpi)$. Note that the core-connectivity-similarity of any two adjacent nodes is equal to their structure similarity when $\mu = 2$, because here the core-similarity of two adjacent nodes is equal to the highest similarity of their incident edges which is not less than their structure-similarity. Hence, the CCMST of a core-connected network is equal to the MST of the original network when $\mu = 2$.

B. Partition Networks on Its CCMST

According to the Cut Property of MST, we can prove that partitioning core-connected network G with ε (remove edge e s.t. $\varpi(e) \leq \varepsilon$ from G) is equal to the partition of its CCMST with ε.

Definition 10. *(Connectivity Level) Let $G = (V, E, \varpi)$ be a core-connected undirected network and each edge $e \in E$ has a value of core-connectivity-similarity $\varpi(e) \in [0, 1]$. Given nodes $u, v \in V$, $u \neq v$ and $\varepsilon \in \mathbb{R}$, u and v are connected if each edge $e \in E$ s.t. $\varpi(e) < \varepsilon$ is removed from G, but they are not connected if each edge $e \in E$ s.t. $\varpi(e) \leq \varepsilon$ is removed from G. Then the connectivity level of u and v in G is ε. This is denoted by $\gamma_G(u, v) = \varepsilon$.*

Theorem 2. *Let $G = (V, E, \varpi)$ be a core-connected undirected network and T be an MST of G. $\forall u, v \in V, u \neq v, \gamma_G(u, v) = \gamma_T(u, v)$.*

Proof: $\forall u, v \in V$, $u \neq v$, let $\gamma_T(u, v) = \varepsilon$, and P be the path between u and v in T. Obviously, $\exists e = \{p, q\} \in P$ *s.t.* $\varpi(e) = \varepsilon$ and ε is the minimal edge weight in P. When each edge d *s.t.* $\varpi(d) < \varepsilon$ is removed from G, path P still remains in G. Thus, nodes u and v will stay connected in G.

Assume that each edge $d \in E$ *s.t.* $\varpi(d) \leq \varepsilon$ is removed from G, u and v are still connected. There must be a path P' between u and v in G s.t. $\forall d \in P'$, $\varpi(d) > \varepsilon$. So $\exists e' = \{p', q'\} \in P' \wedge e' \notin T$, otherwise P and P' will form a cycle in T. $T' = (T - \{e\}) \cup \{e'\}$ is also a Spanning Tree of G s.t. $\varpi(T') > \varpi(T)$. It is inconsistent with that T is a maximal spanning tree of G. So u and v will not be connected when each edge $d \in E$ s.t. $\varpi(d) \leq \varepsilon$ is removed from G. Thus $\gamma_G(u, v) = \gamma_T(u, v) = \varepsilon$. ■

Since each edge of T is a cut edge, different edge weights of T will produce different partition results and form different clusters. Without considering the slight effect of border nodes, all possible ε values lie in the edge weights of the CCMST.

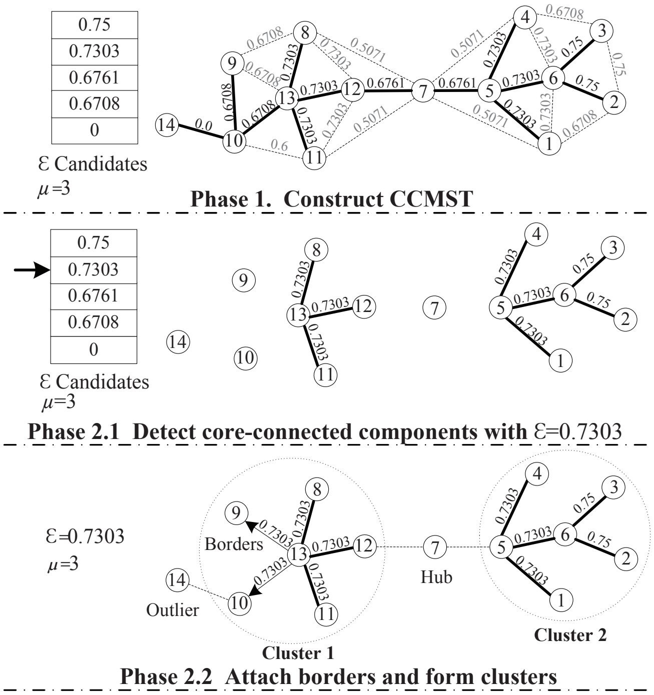


Figure 5. Procedure of the network clustering on the CCMST.

C. Bulid the Attractor Indices

The partition on the CCMST of a network can detect the core nodes for each cluster. In addition, there may exist some border nodes in a cluster. In order to attach the border nodes of each cluster efficiently, we build the indices in advance. The involved concepts are given as follows.

Definition 11. *(Attachability-Similarity) Given $v \in V$, the* attachability-similarity *of v w.r.t. its neighbor nodes is*

$$AS(v) = \max\{RS(u, v) | u \in \Gamma(v) - \{v\}\}. \qquad (6)$$

The attachability-similarity of a node v is the maximum of reachability-similaries w.r.t. its neighbor nodes. The neighbor node that possesses the maximal reachability-similarity to v is regarded as the *attractor* of v.

If $CS(v) < AS(v)$ (i.e., the core-similarity of v is less than its attachability-similarity), then v is not a core node, but its attractor u is a core node when $\varepsilon = AS(v)$. That is to say, v should be attached as a border node to the cluster containing core node u. In this case, an index of the attachability-similarity for node v and its attractor u will be built in advance. As shown in Figure 4(b), the reachability-similarities of node p w.r.t. its neighbor nodes are labeled on the arrowhead lines. Since the value of reachability-similarity from q to p is the highest, node q should be the attractor of p and the attachability-similarity of node p is 0.82 which is equal to the reachability-similarity of p w.r.t q. Note that the attractor can be selected arbitrarily with ties and the indices can be built during the calculation of core-connectivity-similarity.

V. THE ALGORITHM

Our network clustering algorithm on the CCMST includes two phases, and the procedure is illustrated in Figure 5. In the first phase, we calculate the core-connectivity-similarity for each edge in the network, build the attractor indices and construct the CCMST accordingly. After that, we record all the different edge weights of the CCMST as the ε candidates and sort them ascending or descending. In the second phase, we perform the tree clustering on the CCMST. Actually, the clustering results are not sensitive to the parameter μ which can be set as a constant.

There are two different ways to implement the tree clustering. The first is tree agglomerative clustering and the pseudocode is given in Algorithm 1. We consider an initial forest consisting of n isolated nodes. Then we add the edges having the same weights in the CCMST to the forest, from the highest value to the lowest. For each ε, it will agglomerate the nodes into multiple core-connected components after all the edges whose weights are equal to ε have been added to the forest. After attaching the border nodes, we can get the clusters. Then we calculate the Q_s value of current clustering result. The output of the algorithm is the clusters with the highest Q_s value and the corresponding ε.

The second method is tree divisive clustering. We remove the edges in the CCMST w.r.t. different weights or ε, from the lowest value to the highest. For each ε, it will partition the tree into multiple core-connected components. The remaining process is the same as the agglomerative one.

The two tree clustering algorithms above are equivalent in clustering results, but the agglomerative one is convenient for detecting the connected components in the forest. Thus, it is faster than the divisive one. Because the core-connectivity-similarity of any two nodes is equal to their structure-similarity when $\mu = 2$, the procedure of attaching borders can be passed over in this case. Hence, our algorithm contains the traditional MST-based clustering as a special case. We also compared our algorithm with the SCAN algorithm. Though the clustering procedures of gSkeletonClu and SCAN are quite different, the results of these two algorithms are almost the same w.r.t. the same values of parameters ε and μ. In addition, our algorithm can detect overlapping communities by permitting hubs and border nodes to be shared by multiple clusters.

Algorithm 1 gSkeletonClu_Agglomerative

```
Input: Network $G = (V, E, w)$; and $\mu \in \mathbb{N}$
Output: Clusters $CR = \{C_1, C_2, \cdots, C_m\}$; hubs and outliers $N$; and
    optimal $\varepsilon$
//Phase 1: Construct CCMST $T = (V, E')$
Initialize a empty dynamical priority queue $AIQ$;
for each $e = \{u, v\} \in E$ do
    CalculateCCS($u, v, \mu$);
    BuildAttractorIndices($AIQ$, $u$, $v$);
end for
$T = MST(G)$;
$W = \{CCS(e) | e \in E'\}$;
$Sort\_Decend(W)$;
//Phase 2: Tree Clustering
$E^{(0)} = \emptyset$;
$F^{(0)} = (V, E^{(0)})$;
for $i = 1 \ to \ |W|$ do
    //Phase 2.1: Detect connected components
    $\varepsilon^{(i)} = W[i]$;
    $S = \{\{v_i, v_j\} | \{v_i, v_j\} \in E' \wedge CCS(v_i, v_j) = \varepsilon^{(i)}\}$;
    $E^{(i)} = E^{(i-1)} \cup S$;
    $F^{(i)} = (V, E^{(i)})$;
    $CR^{(i)} = \{C[c] | c \in V$, $C[c]$ is a connected component in $F^{(i)}$ $\}$;
    //Phase 2.2: Attach borders
    while $AIQ$.getHead().KeyValue $\geq \varepsilon^{(i)}$ do
        $(p, q, KeyValue)$ = $AIQ$.popHead();
        if $\{\{p\}\} \in CR$ then
            $C[q] = C[q] \cup \{p\}$;
            $CR = CR - \{\{p\}\}$;
        end if
    end while
    //Phase 2.3: Detect clusters, hubs and outliers
    $N^{(i)} = \emptyset$;
    for each $C \in$ $CR^{(i)}$ do
        if $|C| < \mu$ then
            $CR^{(i)} = CR^{(i)} - \{C\}$;
            $N^{(i)} = N^{(i)} \cup C$;
        end if
    end for
    $Q^{(i)} = Q_s(CR^{(i)})$;
end for
$k = \arg\max_t \{Q^{(t)}\}$;
return $CR^{(k)}$, $N^{(k)}$ and $\varepsilon^{(k)}$;
```

The running time of gSkeletonClu is mainly consumed by the construction of CCMST and tree clustering. The core-connectivity-similarity and attractor indices can be calculated within a complexity of $O(m)$. We construct the CCMST of the core-connected network using the Prim's algorithm with a Fibonacci Heap. Its running time complexity is $O(m \log n)$. In the procedure of agglomerative clustering, we only need to add the edges to the forest in order of weights with a complexity of $O(n \log n)$. However, the calculation of Q_s is a little time consuming. We implement it in an incremental way and the complexity is $O(m)$ when we try all possible ε. In total, the time complexity of our algorithm is $O((m+n) \log n)$. For the scale-free networks, it is $O(m \log n)$. The algorithm can also be implemented in a more efficient way, which starts the clustering process from the 1/2 of ordered edges and stops when the current Q_s value decreases by 0.05 from the previous one. The optimized algorithm reduces almost 1/2 running time with the same clustering results.

VI. EXPERIMENTS

In this section, we evaluate the proposed algorithm using some real-world networks and synthetic datasets. The performance of gSkeletonClu is compared with two state-of-the-art methods: SCAN and FastModularity. SCAN is an efficient density-based network clustering algorithm, and FastModularity is a representative modularity-based algorithm for community detection. Our algorithm is implemented using

Table I
THE PARAMETERS OF THE COMPUTER-GENERATED DATASETS FOR PERFORMANCE EVALUATION.

Dataset	n	m	k	$maxk$	$minc$	$maxc$
10000S	10,000	99,238	20	50	10	50
10000B	10,000	99,036	20	50	20	100
100000S	100,000	1,987,850	40	100	50	100
100000B	100,000	1,990,271	40	100	100	200

Table II
THE OPTIMAL ε AND THE CORRESPONDING Q_s.

Dataset	Manually selected parameter ε		Global optimal parameter ε	
	ε	Q_s	ε	Q_s
football	0.5466	0.7231	0.5222	0.7622
polbooks	0.4376	0.5532	0.3746	0.5645
10000S	0.2678	0.7964	0.2101	0.8554
10000B	0.1866	0.7426	0.1909	0.7715
100000S	0.2451	0.8077	0.1636	0.8408
100000B	0.1593	0.8699	0.1319	0.8828

ANSI C++. All the experiments were conducted on a PC with a 2.4 GHz Pentium IV processor and 4GB of RAM.

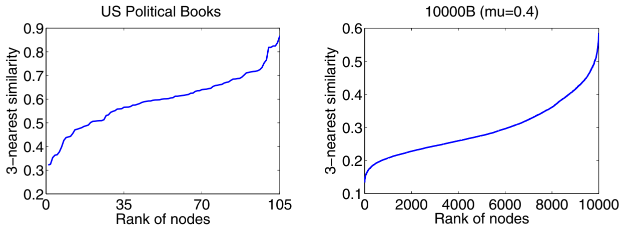
A. Datasets

We evaluate the performance of our algorithm on two types of datasets. One is the real-world networks and the other is the computer-generated benchmark networks with known community structure.

1) Real-world Networks: To assess the performance of the proposed method in terms of accuracy, we conduct experiments on two popular real-world networks: NCAA College-football network and US Political Books network [20]. The NCAA College-football is a social network with communities (or conferences) of American college football teams. The network, representing the schedule of Division I-A games for the 2000 season, contains 115 nodes and 613 edges. All college football teams are divided into eleven conferences and five independent teams (Utah State, Navy, Notre Dame, Connecticut and Central Florida) that do not belong to any conference. There is a link between two teams if they played a game together. Now the question is to find out the communities from the graph that represents the schedule of games played by all teams. The network of US Political Books contains 105 nodes and 441 edges. The nodes of the network represent the US political books sold by the online bookseller Amazon.com and have been given labels “liberal”, “neutral”, or “conservative”, respectively, by Newman. There is a link between two books if they are co-purchased frequently enough by the same buyers.

2) Synthetic Benchmark Networks: We also use the Lancichinetti-Fortunato-Radicchi (LFR) benchmark graphs [21] to evaluate the performance of our algorithm. By varying the parameters of the networks, we analyze the behavior of the algorithms in detail. Some important parameters of the benchmark networks are:

- n: number of nodes
- m: average number of edges
- k: average degree of the nodes
- $maxk$: maximum degree
- mu: mixing parameter, i.e., each node shares a fraction mu of its edges with nodes in other communities
- $minc$: minimum for the community sizes
- $maxc$: maximum for the community sizes



(a) (b)

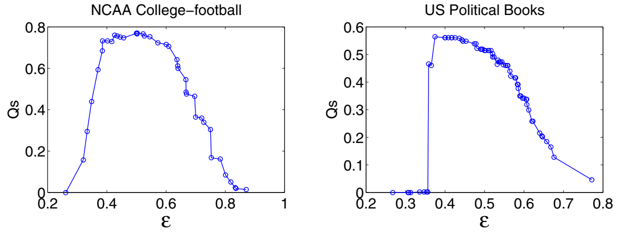
Figure 6. Sorted 3-nearest structural similarity.

We generate several weighted undirected benchmark networks with the number of nodes $n =$ 10,000 and 100,000. In Table I, we list the values of the parameters for the generated datasets. For each n, two types of networks are generated with different ranges of the community sizes, where S means that the sizes of the communities in the dataset are relatively small and B means that the sizes of the communities are relatively big. For each type of datasets, we generate fifteen networks with different mixing parameter mu ranging from 0.1 to 0.8 with a span of 0.05. Generally, the higher the mixture parameter of a network is, the more difficult it is to reveal the community structure.

B. Selection of the Parameter ε

In [2], the authors presented a “knee hypothesis” to find the proper ε value. We sort 3-nearest similarity of all nodes in the networks to locate the knee. Two plots for the US Political Books and the benchmark 10000B are given respectively in Figure 6, and it can be observed that there are no obvious knee in the curves. Furthermore, there is no rigorous way to ensure that the identified “knees” are the appropriate values of the parameter ε.

In the experiment, we try our best to select a “knee” as the parameter ε for the SCAN algorithm from the 3-nearest similarity plot of each network. In contrast, our algorithm locate the optimal ε automatically which achieves the highest Q_s value. In Table II, we show the manually selected values of ε and the corresponding Q_s for some adopted datasets, as well as the optimal ε and the corresponding Q_s values found by gSkeletonClu, where $mu = 0.4$ for all of the benchmark networks. As shown in Table II, the manually selected ε is



(a) (b)

Figure 7. The distribution of Q_s w.r.t. ε.

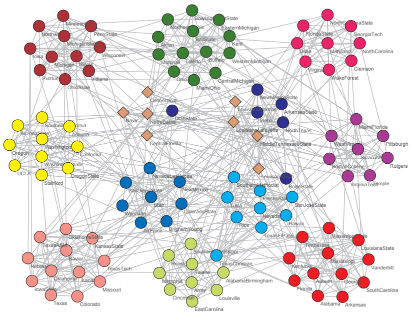


Figure 8. NCAA College-football network.

always not the optimal one because it depends greatly on the intuition of user.

We also present the relationship between ε and Q_s in Figure 7. It can be observed that the maximum of Q_s in the curve is always near the middle of all possible ε values. Thus we can locate the optimal ε by testing only a portion of all possible values. We introduce two parameters for our algorithm: b and e. The parameter b indicates the start position of all the sorted edges in the CCMST and the parameter e indicates the stop criteria. For example, if we set $b = 0.5$ and $e = 0.05$, it means that the clustering process will start from a half of all edges with higher weights which have been appended in the forest and stop when the current Q_s value decreases by 5% from the previous one.

C. Criteria for Accuracy Evaluation

In our experiments, we adopt Normalized Mutual Information (NMI) [22], an information-theoretic based measurement, to evaluate the quality of clusters generated by different methods. This is currently widely used in measuring the performance of network clustering algorithms. Formally, the measurement metric NMI can be defined as

$$NMI = \frac{-2\sum_{i,j} N_{ij} log(\frac{N_{ij}N}{N_{i.}N_{.j}})}{\sum_i N_{i.}log(\frac{N_{i.}}{N}) + \sum_j N_{.j}log(\frac{N_{.j}}{N})}, \qquad (7)$$

where N is the confusion matrix, N_{ij} is the number of nodes in both X_i and Y_j, $N_{i.}$ is the sum over row i of N and $N_{.j}$ is the sum over column j of N. Note that the NMI value ranges between 0.0 (total disagreement) and 1.0 (total agreement).

D. Accuracy Comparison on Real-world Networks

NCAA College-football network: Figure 8 illustrates the original NCAA College-football network and the clustering result of our algorithm with each node representing a school team. For the original network, the conferences and the group of five independent teams are indicated by cliques. Our algorithm obtains eleven clusters in this network which demonstrates a perfect match with the original conference system. The teams belonging to a conference and the independent teams are denoted by circles and diamonds respectively, and the teams in the same conferences are represented by the same color. Four independent teams are correctly identified as hubs. Although another four teams (i.e., Louisiana Monroe, Louisiana Lafayette, Louisiana Tech, and Middle Tennessee State) are identified as hubs, and three other teams are misclassified, our algorithm still performs much better than other methods including SCAN and FastModularity, which will be described as follows.

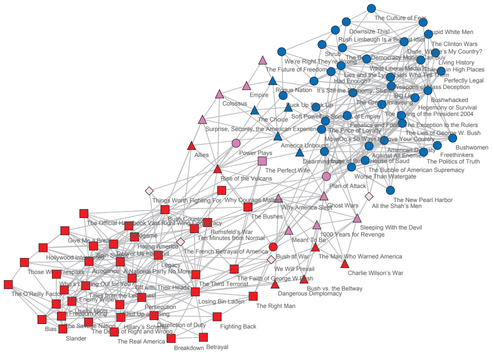


Figure 9. US Political Books network.

The SCAN algorithm detects thirteen communities as its best result in this dataset with parameters ($\varepsilon = 0.5466$, $\mu = 3$). The teams of two conferences are divided into two clusters respectively. Meanwhile, it finds ten hub nodes with only four are correctly identified, and one independent team UtahState is misclassified into a conference. It follows that the parameter value selected manually lowers the accuracy of SCAN. The modularity-based algorithm FastModularity discovers seven communities, but only four communities matching with the conferences. For the five independent teams, they are assigned to three different communities.

US Political Books network: The original network and the clustering result of our algorithm are presented in Figure 9. For the original network, the "liberal", "neutral" and "conservative" books are represented by circle, triangle and box, respectively. Our algorithm successfully identifies three clusters and four hubs with $\mu = 4$. The clusters detected by our algorithm are illustrated by three different colors: blue for "liberal" books, thistle for "neutral" books and red for "conservative" books. In addition, the hubs which do not belong to any cluster are denoted by pink diamonds.

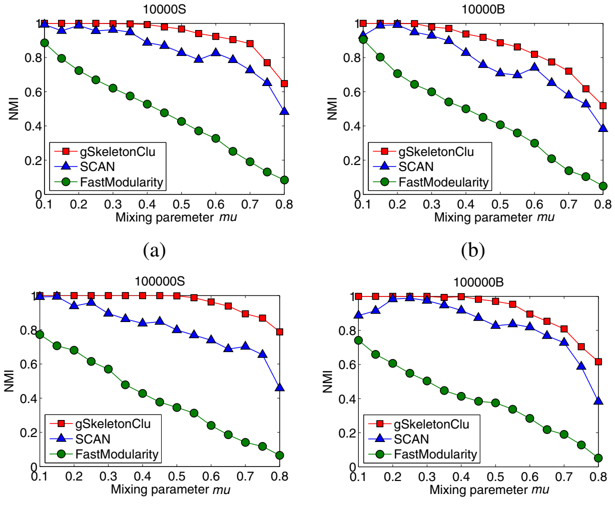
The SCAN algorithm detects three clusters and nine hubs in the network with parameters ($\varepsilon = 0.4376, \mu = 4$). The FastModularity algorithm finds four clusters in this network, and even worse it mixes some nodes from all the three communities together as a new cluster.

In summary, gSkeletonClu generates promising clustering results along with hubs and outliers in community detection, consistently outperforming baseline methods including SCAN and FastModularity.

E. Accuracy Comparison on Synthetic Networks

For a more standardized comparison, we turn to the recently proposed LFR benchmark graphs, which are claimed to possess properties found in real-world networks and incorporate more realistic scale-free distributions of node-degree and cluster-size [21]. We compare the three clustering algorithms on the networks with size of 10,000 and 100,000.

The NMI scores of the three methods are plotted in Figure 10. On most of the datasets, our algorithm gets



(a) (b)

(c) (d)

Figure 10. NMI values of the three algorithms on the computer-generated benchmark networks.

$NMI = 1$ as $mu < 0.4$, which means a perfect match with the original network structure. However, the performance of our algorithm decreases when $mu > 0.4$, especially in the small-scale network with big communities (e.g., 10000B). This is because that more and more hubs and outliers are isolated with the increasing of parameter mu. As shown in Figure 10, the performance of gSkeletonClu is better than that of FastModularity on all generated networks, because the FastModularity algorithm tends to produce a small number of big communities on the large-scale networks, due to the well known resolution limit of modularity [19].

For SCAN, its NMI values are lower than that of our algorithm but higher than FastModularity in most cases. This verifies the advantage of the density-based methods in community detection within complex network. However, we observe that the clustering results of SCAN are sometimes unreasonable. For example, the NMI value of SCAN on the network with $mu = 0.6$ is higher than that with $mu = 0.5$ in Figure 10(b), and the NMI value of SCAN on the network with $mu = 0.1$ is much lower than that with $mu = 0.2$ in Figure 10(d). This is in conflict with the characteristic of the LFR benchmark networks, which demonstrates that SCAN is sensitive to the parameter ε and the manually selected parameter can not provide the optimal clustering results.

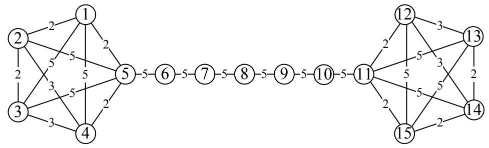
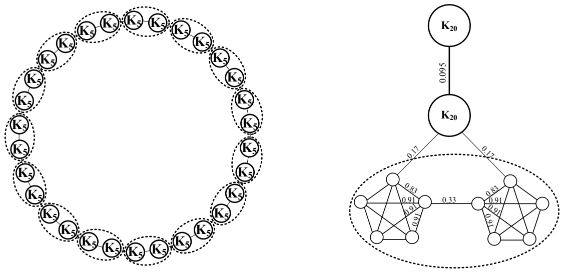


Figure 11. The Dumbbell schematic networks (the numbers on the edge represent the weights).



(a) (b)

Figure 12. Two schematic networks (the numbers on the edge represent the structural similarity): (a) the Ring network made out of identical cliques connected by single links, and (b) the Pairwise network with four cliques.

The NMI values clearly demonstrate that gSkeletonClu can always locate the optimal ε and produce clusters that resemble the true classes of the datasets in our study. When we use the optimal ε found by our algorithm as the input parameter for SCAN, it can get the same clustering results in most cases. But there will still be some minor differences because our algorithm assigns the border node to its neighbor core node with the maximum of reachability-similarity, while SCAN assigns the border node to the core node which associates with it first.

F. Tackling the Problem of Chaining Effect

The traditional MST clustering method may suffer from the problem of chaining effect (also called as single-link effect), which is caused by some linearly connected nodes that run through a sparse area. In Figure 11, we present a network, called Dumbbell, which consists of two cliques with five nodes connected by another five nodes in single links. If we use the MST clustering approach to cluster the original weighted Dumbbell network, we can get only one cluster containing all the nodes in the network. The reason is that the weights of the edges in the MST are all equal to five. But if we use the MST clustering algorithm smoothed by local similarity (i.e., gSkeletonClu with $\mu = 2$), we get three clusters: {1,2,3,4,5}, {6,7,8,9,10}, and {11,12,13,14,15}. Interestingly, our algorithm gSkeletonClu obtains the most reasonable clustering results with $\mu = 4$ or 5, in which the two density cliques are discovered as two clusters and the middle five nodes connected by single links are identified as outliers.

Obviously, our algorithm can overcome the chaining effect because it uses structure similarity as edge weight, which is smoothed by local link-density. Moreover, the parameter μ can be used to determine the minimal size of clusters freely.

G. Analyzing the Problem of Resolution Limit

Despite the good performance of the modularity measure on many practical networks, it may lead to apparently unreasonable partitions in some cases. It has been shown that modularity contains an intrinsic scale that depends on the total number of links in the network. Communities that are smaller than this intrinsic scale may not be resolved, even in the extreme case that they are complete graphs connected by some single bridges. The resolution limit of modularity actually depends on the degree of interconnectedness between pairs of communities and can reach values of the order of the size for the whole network [19].

Table III
THE NUMBER OF COMMUNITIES ON RING AND PAIRWISE NETWORKS FOUND BY SA, FASTMODULARITY AND GSKELETONCLU.

Dataset				SA	FastModularity	gSkeletonClu
Name	N	M	C			
Ring	150	330	30	15	16	**30**
Pairwise	50	404	4	3	3	**4**

In Figure 12(a), we show a network consisting of a ring of several cliques, connected through single links. Each clique is a complete graph with n nodes and $n(n-1)/2$ links. Suppose there are c cliques (with c even), the network has a total of $N = nc$ nodes and $M = cn(n-1)/2 + c$ edges. According to [19], modularity optimization would lead to a partition where the cliques are combined into groups of two or more (represented by dotted lines). Here, we use a synthetic dataset with $n = 5$ and $c = 30$, called Ring. Another synthetic network is shown in Figure 12(b). In this network, the larger circles represent cliques with n nodes, denoted as K_n, and there are two small cliques with p nodes. According to [19], we set $n = 20$, $p = 5$ and obtain the network called Pairwise. Modularity optimization merges the two small communities into one (shown with a dotted line).

We present the clustering results on the above two datasets in Table III, where N is the number of nodes, M is the number of edges, and C is the correct number of communities. Our algorithm gSkeletonClu discovers the exact communities. For the Ring and Pairwise networks, the modularity-based algorithms SA and FastModularity both possess the resolution limit problem which results in merging two small cliques into one cluster.

The reason that our algorithm can overcome the resolution limit is that it combines the density-based clustering principle and the modularity measure. The connected nodes with higher similarity will be considered preferentially as in the same community than the lower ones. Moreover, all of the adjacent nodes with equal similarities will be merged in one community together or be staying alone.

VII. CONCLUSIONS

In this paper, a novel network clustering algorithm, called gSkeletonClu, is presented to overcome the sensitive parameter problem of density-based network clustering and detect clusters, hubs and outliers in large-scale undirected networks. By projecting the original weighted network to its CCMST, we convert the problem of community discovery to finding core-connected components in the CCMST. A theoretical analysis shows that the structural clustering result on the CCMST is equal to that on the original network. Our algorithm also overcomes the problem of chaining effect possessed by the traditional MST-based clustering methods and the resolution limit possessed by other modularity-based algorithms. In the future, it is interesting to use our method to analyze the database data and more complex graphs from various applications.

ACKNOWLEDGMENTS

The work was supported in part by the Natural Science Basic Research Plan in Shaanxi Province of China (No. SJ08-ZT14), the National High-Tech Research and Development Plan of China under Grant No.2008AA01Z131, the U.S. National Science Foundation grants IIS-08-42769, CCF-0905014, and BDI-07-Movebank, and the Air Force Office of Scientific Research MURI award FA9550-08-1-0265. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," in *Proc. of SODA'98*, 1998, pp. 668–677.

[2] X. Xu, N. Yuruk, Z. Feng, and T. Schweiger, "SCAN: a structural clustering algorithm for networks," in *KDD'07*. ACM, 2007, pp. 824–833.

[3] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD'96*, vol. 96, 1996, pp. 226–231.

[4] D.-H. Kim, J. D. Noh, and H. Jeong, "Scale-free trees: The skeletons of complex networks," *Phys. Rev. E*, vol. 70, no. 4, p. 046126, Oct 2004.

[5] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, no. 1, pp. 291–307, 1970.

[6] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, 1998.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE TPAMI*, vol. 22, no. 8, pp. 888–905, 2000.

[8] S. White and P. Smyth, "A spectral clustering approach to finding communities in graph," in *SDM'05*, 2005.

[9] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *SIGMOD'99*, 1999, pp. 49–60.

[10] D. Bortner and J. Han, "Progressive clustering of networks using structure-connected order of traversal," in *ICDE'10*, 2010.

[11] C. Zahn, "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters," *IEEE Transactions on Computers*, vol. 100, no. 20, pp. 68–86, 1971.

[12] J. C. Gower and G. J. S. Ross, "Minimum spanning trees and single linkage cluster analysis," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 18, no. 1, pp. 54–64, 1969.

[13] Y. Xu, V. Olman, and D. Xu, "Minimum spanning trees for gene expression data clustering," *Genome Informatics*, vol. 12, p. 2001, 2001.

[14] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Clustering validity checking methods: part I," *SIGMOD Rec.*, vol. 31, no. 2, pp. 40–45, 2002.

[15] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, no. 2, p. 026113, 2004.

[16] Z. Feng, X. Xu, N. Yuruk, and T. A. J. Schweiger, "A novel similarity-based modularity function for graph partitioning," in *DaWaK'07*, 2007, pp. 385–396.

[17] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, p. 066111, Dec 2004.

[18] R. Guimera and L. N. Amaral, "Functional cartography of complex metabolic networks," *Nature*, vol. 433, no. 7028, pp. 895–900, 2005.

[19] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *PNAS*, vol. 104, no. 1, p. 36, 2007.

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

[21] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E*, vol. 78, no. 4, p.046110, 2008.

[22] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, p. P09008, 2005.