Revealing Density-Based Clustering Structure from the Core-Connected Tree of a Network

Jianbin Huang, Heli Sun, Qinbao Song,
Hongbo Deng, *Member*, *IEEE*, and Jiawei Han, *Fellow*, *IEEE*

Abstract—Clustering is an important technique for mining the intrinsic community structures in networks. The density-based network clustering method is able to not only detect communities of arbitrary size and shape, but also identify hubs and outliers. However, it requires manual parameter specification to define clusters, and is sensitive to the parameter of density threshold which is difficult to determine. Furthermore, many real-world networks exhibit a hierarchical structure with communities embedded within other communities. Therefore, the clustering result of a global parameter setting cannot always describe the intrinsic clustering structure accurately. In this paper, we introduce a novel density-based network clustering method, called graph-skeleton-based clustering (gSkeletonClu). By projecting an undirected network to its core-connected maximal spanning tree, the clustering problem can be converted to detect core connectivity components on the tree. The density-based clustering of a specific parameter setting and the hierarchical clustering structure both can be efficiently extracted from the tree. Moreover, it provides a convenient way to automatically select the parameter and to achieve the meaningful cluster tree in a network. Extensive experiments on both real-world and synthetic networks demonstrate the superior performance of gSkeletonClu for effective and efficient density-based clustering.

Index Terms—Network clustering, density-based method, hierarchical clustering, spanning tree, parameter selection

1 Introduction

Network is a widespread form of data consisting of a set of vertices and a set of edges that are connections between pairs of vertices. Many real-world networks possess an intrinsic community structure, such as large social networks, Web graphs, and biological networks. A community (also referred to as cluster) is typically thought of as a group of vertices with dense connections within groups and relatively sparse connections between groups as well. Since clustering is an important technique for mining the intrinsic community structure in networks, it has become an important problem in a number of fields, ranging from social network analysis to image segmentation and from analyzing biological networks to the circuit layout problem [2].

However, detecting communities in complex networks is a nontrivial task, since the structure of a large-scale network is highly complex and the communities are in arbitrary size and shape. Moreover, it has been shown that there always exits a hierarchical structure in many complex networks with communities embedded within other communities [3]. Essentially, small communities group together to form larger ones, which in turn group together to form even larger ones. Besides the cluster vertices densely connected within communities, there are some vertices 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 [4], and hubs in viral marketing and epidemiology could be central nodes for spreading ideas or diseases [5]. On the contrary, outliers are marginally connected with the community members which should be isolated as noises. Therefore, how to detect multilevel communities and identify hubs and outliers in a network has become an interesting and challenging problem.

A recently proposed network clustering algorithm SCAN [6] extended from the traditional density-based clustering algorithm DBSCAN [7] not only detects meaningful clusters, but also identifies the vertices in special roles, such as hubs and outliers. 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 hard to determine. Actually, how to select the parameter ε automatically for the density-based clustering methods (e.g., DBSCAN and SCAN) is a long-standing and challenging task. Moreover, SCAN searches for the clusters in a network by using a global parameter setting and has difficulty to handle the hierarchical clustering structure. To deal with the problem, Bortner and Han proposed an algorithm, called SCOT+HintClus [8], to detect the hierarchical cluster boundaries of a network through extension of the algorithm OPTICS [9]. However, it has difficulty to locate the global optimal ε and to extract the meaningful cluster tree.

- J. Huang is with the School of Software, Xidian University, PO Box 61, No. 2, South Taibai Road, Xi'an, Shannxi 710071, China. E-mail: jbhuang@xidian.edu.cn.
- H. Sun and Q. Song are with the Department of Computer Science and Technology, Software Institute in the School of Electronic and Information Engineering, Xi'an Jiaotong University, No. 28, West Xianning Road, Xi'an, Shannxi 710049, P.R. China. E-mail: {hlsun, qbsong}@mail.xjtu.edu.cn.
- H. Deng and J. Han are with the Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N. Goodwin Ave., Siebel Center, Urbana, IL 61801. E-mail: hbdeng@uiuc.edu, hanj@cs.uiuc.edu.

Manuscript received 12 Apr. 2011; revised 14 Sept. 2011; accepted 24 Apr. 2012; published online 9 May 2012.
Recommended for acceptance by J. Sander.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-04-0199.
Digital Object Identifier no. 10.1109/TKDE.2012.100.

We have found that each density-based cluster can be determined by a set of cores (a special class of vertices that have a minimum of μ neighbors with a structural similarity exceeding the threshold ε [6]) embedded in it uniquely. Hence, once all the components of connected cores have been detected, all clusters can be obtained. Accordingly, the problem of detecting clusters for a certain value of parameter ε in an undirected network can be converted to find core-connected components in it. A core-connected maximal spanning tree (CCMST) is prepared in advance which contains the necessary information for extracting the density-based clustering for various values of ε. Then a novel density-based clustering algorithm, called graph skeleton-based clustering (gSkeletonClu), is proposed which can perform the clustering on the spanning tree of the original network. It can select the parameter ε automatically by using a cluster validity measure. Furthermore, meaningful hierarchical clusters also can be easily revealed by our algorithm, because density-based clustering with respect to various values of parameter ε can be detected progressively on the tree. Experimental results indicate that our algorithm is scalable and achieves high accuracy.

The remainder of the paper is organized as follows: Section 2 introduces the concepts of structural connected clusters. Section 3 formalizes the notion of clusters derived from the core-connected components and presents a theoretical analysis. Section 4 describes the algorithm in detail. In Section 5, the experimental results are reported. Section 6 reviews the related literature. Finally, Section 7 concludes the paper.

2 Preliminaries

Let $G = (V, E, w)$ be a weighted undirected network, where V is a set of vertices, E a set of edges and w a function that assigns each edge a positive number as its weight. Our goal is to detect all the clusters and identify hubs and outliers in it. Therefore, both local structure and connectivity are used in our definition of network clustering. In this section, we formalize the notion of a structure-connected cluster in networks, which extends that of a density-based cluster [6], [7].

The structure of a vertex can be described by its neighborhood, and two vertices are assigned to a cluster always according to how they share neighbors. A formal definition of structure neighborhood is given as follows:

Definition 1 (Structure Neighborhood). *The structure neighborhood of a vertex $u \in V$ is the set $\Gamma(u)$ containing u and its adjacent vertices which are incident to a common edge with u:*

$$\Gamma(u) = \{v \in V | \{u, v\} \in E\} \cup \{u\}. \tag{1}$$

Here, a similarity function can be employed to measure the local connectivity density of any two adjacent vertices in a network. A type of widely used similarity function is based on common neighborhood (e.g., Cosine and Jaccard) [10]. In this paper, we introduce a structural similarity which is defined as

$$\sigma(u, v) = \frac{\sum_{x \in \Gamma(u) \cap \Gamma(v)} w(u, x) \cdot w(v, x)}{\sqrt{\sum_{x \in \Gamma(u)} w^2(u, x)} \cdot \sqrt{\sum_{x \in \Gamma(v)} w^2(v, x)}}. \tag{2}$$

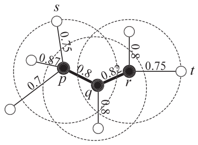


Fig. 1. The relationship of structure reachability and structure connectivity between the vertices in a density-based cluster.

It is able to deal with weighted undirected network which is extended from the cosine similarity used in [6]. The more common neighbor vertices a vertex shares with a neighbor vertex of it, the greater the structural similarity between them is. Then we apply a threshold ε for assigning cluster membership.

Definition 2 (ε-Neighborhood). *Let $u \in V$ and σ be a similarity function for any pair of adjacent vertices, and $\varepsilon \in \mathbb{R}$ be a threshold of similarity. The set of neighbors of u having structural similarity no less than ε forms its ε-neighborhood*

$$\Gamma_\varepsilon(u) = \{v \in \Gamma(u) | \sigma(u, v) \geq \varepsilon\}. \tag{3}$$

Definition 3 (Core). *Let $\mu \in \mathbb{N}$. For a vertex u, if $|\Gamma_\varepsilon(u)| \geq \mu$, then u is a core, denoted by $\mathrm{K}_{\varepsilon,\mu}(u)$.*

Cores are a class of vertices that have at least μ neighbors possessing the structural similarity no less than a threshold ε. In this way, the parameters ε and μ determine the clustering of a network. As shown in Fig. 1, when we set $\varepsilon = 0.75$ and $\mu = 4$, the vertex p is a core and the vertex q is in the ε-neighborhood of p. According to the principle of density-based clustering, the clusters grow from core vertices. If a vertex is in the ε-neighborhood of a core, it should be in the same cluster with the core. This idea is formalized in the following definition of directly structure-reachable.

Definition 4 (Directly Structure-Reachable). *Given $\varepsilon \in \mathbb{R}$ and $\mu \in \mathbb{N}$, a vertex $v \in V$ is directly structure-reachable from a vertex $u \in V$ iff $\mathrm{K}_{\varepsilon,\mu}(u) \wedge v \in \Gamma_\varepsilon(u)$, denoted by $u \mapsto_{\varepsilon,\mu} v$.*

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

Definition 5 (Structure-Reachable). *A vertex $v \in V$ is structure-reachable from $u \in V$ iff $\exists \{u_1, \ldots, u_n\} \subseteq V s.t.$ $u = u_1, v = u_n,$ and $\forall i \in \{1, 2, \ldots, n-1\}$ such that $u_i \mapsto_{\varepsilon,\mu} u_{i+1}$. This is denoted by $u \rightarrow_{\varepsilon,\mu} v$.*

We depict the notion of structure-reachable in Fig. 1. When we set $\varepsilon = 0.75$ and $\mu = 4$, there are four vertices in the neighborhood set of vertices $p, q,$ and r, respectively, with structure similarities no less than current ε, thus $p, q,$ and r are all core vertices. Since the structure similarity between p and q is greater than 0.75, vertices 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 5, the cores $p, q,$

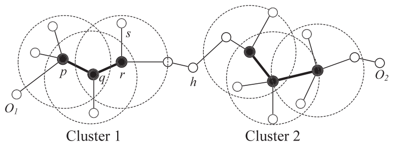


Fig. 2. An example graph with two structure-connected clusters, one hub and two outliers.

and r are structure-reachable from each other. However, the border s is only structure-reachable from the above three cores on one side.

The transitive closure of the directly structure-reachable relation forms clusters which is formally defined in the following, and any pair of vertices in the same cluster is *structure connected*.

Definition 6 (Structure-Connected Cluster). *The set* $C[u] \subseteq V$ *is a cluster represented by* $\mathrm{K}_{\varepsilon,\mu}(u) \in V$ *iff 1)* $u \in C[u]$*; and 2)* $\forall v \in V$, $u \rightarrow_{\varepsilon,\mu} v \Leftrightarrow v \in C[u]$.

Actually, a structure-connected cluster is uniquely determined by the structure-connected cores in it. Each cluster C has at least one core and it contains exactly the vertices which are structure-reachable from an arbitrary core in C. That is to say, the size of each cluster is not less than the value of prespecified parameter μ. If there are two cores $u, v \in V s.t.\ u \rightarrow_{\varepsilon,\mu} v$, then $C[u] = C[v]$. As shown in Fig. 2, cluster 1 contains all the vertices which are structure-reachable from cores p, q, and r. Thus, $C[p] = C[q] = C[r]$. Besides the cores, there also may exist some border vertices in a cluster (e.g., vertex s in Fig. 2). Note that a border may be structure-reachable from cores in different clusters and thus it may be adopted by one or more clusters.

Given parameters $\varepsilon \in \mathbb{R}$ and $\mu \in \mathbb{N}$, the clustering $CR_{\varepsilon,\mu}$ of a network G consists of all structure-connected clusters in G. After the network is clustered, there may be some vertices that are not suitable to be assigned to any clusters and they may be hubs or outliers.

Definition 7 (Hub and Outlier). *Given a clustering* $CR_{\varepsilon,\mu}$ *of network G, a vertex* $h \in V$ *is a hub iff 1) h does not belong to any cluster:* $\forall C \in CR_{\varepsilon,\mu}$, $h \notin C$*; 2) h bridges multiple clusters:* $\exists C, D \in CR_{\varepsilon,\mu}$, $C \neq D$, *and* $\exists u \in C, v \in D$, *s.t.* $h \in \Gamma(u) \wedge h \in \Gamma(v)$. *If a vertex* $o \in V$ *does not belong to any cluster and is not a hub, it is an outlier.*

As shown in Fig. 2, the vertex h that connects two vertices from different clusters is regarded as a hub, and the vertices o_1 and o_2, which are connected with only one cluster, should be identified as outliers. Since outliers have little or no influence in the clustering of a network, they should be isolated as noises.

3 Clusters Derived from Core-Connected Components

For a specific parameter setting, the density-based clustering algorithm SCAN searches for structure-connected clusters and isolates hubs and outliers in a network by visiting each vertex only once. However, the algorithm SCAN is sensitive to the parameter ε. It often produces very different clustering results even for slightly different parameter settings. In this section, we propose a novel approach for revealing density-based clustering structure in a network by detecting core-connected components of each cluster in various ε levels. It can be easily performed on the spanning tree of the network. The involved concepts are introduced in the following.

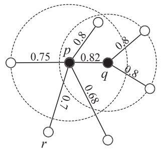


Fig. 3. An example graph for calculating core-similarity, reachability-similarity, and core-connectivity-similarity.

3.1 Core Connectivity Similarity

For detecting cores in each cluster, we first need to determine whether each vertex $u \in V$ is a core under a specific parameters setting.

Definition 8 (Structure Core-Similarity). *Given a vertex* $u \in V$*, the structure core-similarity of u is*

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

The core-similarity of a vertex u is the maximum of structural-similarity $\tilde{\varepsilon}$ such that u would be a core vertex with respect to $|\Gamma_{\tilde{\varepsilon}}(u)| \geq \mu$. Otherwise, the core-similarity is zero. As shown in Fig. 3, the core-similarity of vertex p is 0.75 when $\mu = 4$, because the size of the ε-Neighborhood of vertex p is just four when $\varepsilon = 0.75$ and p will no longer be a core when $\varepsilon > 0.75$. Just the same, the core similarity of vertex q is 0.8.

Definition 9 (Reachability-Similarity). *Given vertices u,* $v \in V$*, the reachability-similarity of v w.r.t. u is*

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

The above two concepts are extended from the literature [9]. Intuitively, the reachability-similarity of a vertex v w.r.t. a vertex u is the maximum of structural similarity such that v is directly structure-reachable from u when u is a core. As shown in Fig. 3, the reachability-similarity of vertex q w.r.t. the core vertex p is 0.75 which is equal to the core-similarity of p, because the similarity between the two vertices is greater than the core-similarity of p. While the reachability-similarity of the vertex r w.r.t. p is 0.7 which is equal to the similarity between vertices p and r. If the value of the parameter ε is not more than $RS(u,v)$, vertex u is a core and vertex v is a member of cluster $C[u]$. However, the reachability-similarity is an asymmetric measurement. Therefore, if we detect clustering through reachability between vertices by visiting each vertex in a network once, the clustering result may be related to the traversing order of the vertices and some border vertices may be misclassified [6], [8], [9].

To overcome the problem above, our method only detects the connected cores in a cluster and deals with the border vertices using an additional process. So it is necessary to determine that whether two vertices u and v

are both cores and are reachable from each other w.r.t. certain values of parameters ε and μ. If so, u and v will lie in the same cluster. Then we introduce a symmetric concept of core connectivity in the following.

Definition 10 (Structure 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 $u \leftrightarrow_{\varepsilon,\mu} v$.*

If two vertices are both cores and 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 vertices in the same closure is *core-connected*.

Definition 11 (Core-Connectivity-Similarity). *Given $\{u, v\} \in E$, the core-connectivity-similarity of u and v is*

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

The core-connectivity-similarity of two vertices is the minimum of their core-similarities and the structure-similarity between them. As shown in Fig. 3, the vertices p and q are core-structure-connected when $\varepsilon = 0.75$ and $\mu = 4$, because p and q are both cores and their structure-similarity is not less than the current ε. But if we set current $\varepsilon > 0.75$, vertices p and q are not core-structure-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 vertices is the maximal threshold of structural similarity such that they are both cores and directly structure-reachable from each other.

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

Proof. Assume $\hat{\varepsilon} > 0$, then $|\Gamma_{\hat{\varepsilon}}(u)| \geq \mu$ and $|\Gamma_{\hat{\varepsilon}}(v)| \geq \mu.\hat{\varepsilon} = CCS(u, v) = \min\{CS(u), CS(v), \sigma(u, v)\}$.

1. If $\varepsilon \leq \hat{\varepsilon}$, then $\varepsilon \leq \min\{CS(u), CS(v), \sigma(u, v)\}$. So $\varepsilon \leq CS(u)$, $\varepsilon \leq CS(v)$, and $\varepsilon \leq \sigma(u, v)$. u and v are both core vertices, and $u \mapsto_{\varepsilon,\mu} v$, then $u \leftrightarrow_{\varepsilon,\mu} v$.
2. If $\varepsilon > \hat{\varepsilon}$, then $\varepsilon > \min\{CS(u), CS(v), \sigma(u, v)\}$. So $\varepsilon > CS(u)$, $\varepsilon > CS(v)$, or $\varepsilon > \sigma(u, v)$. That is, u or v is not a core, or $v \notin \Gamma_\varepsilon(u)$. Thus $u \not\leftrightarrow_{\varepsilon,\mu} v$. □

For each edge $e = \{u, v\} \in E$ in an undirected network $G = (V, E)$, we calculate the core-connectivity-similarity for each pair of adjacent vertices u and v. Let $\varpi(e) = CCS(u, v)$, we will get a core-connected weighted network $G = (V, E, \varpi)$. Then we build the CCMST on $G = (V, E, \varpi)$. The core-connected components can be easily detected on the CCMST of a network. Since the core-similarity of the two adjacent vertices is equal to the largest similarity of their incident edges which is no less than their structural similarity $\sigma(u, v)$ when $\mu = 2$. The core-connectivity-similarity of any two adjacent vertices is equal to their structural similarity in this case. Therefore, the CCMST of a network is equal to the MST of the network weighted by the structural similarity when $\mu = 2$.

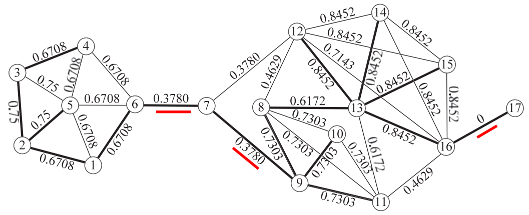


Fig. 4. A sample network weighted by core-connected similarity with parameter $\mu = 4$ which possesses clusters of different size and density.

3.2 Core-Connected Components on the CCMST

According to the Cut Property of MST [11], we can prove that partitioning the weighted network $G = (V, E, \varpi)$ with ε (remove edge e s.t. $\varpi(e) < \varepsilon$ from G) is equal to the partition of its CCMST with ε. Therefore, the core-connected components can be easily detected on the CCMST, the connectivity skeleton of the network [12].

Definition 12 (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 vertices $u, v \in V$, $u \neq v$, and $\varepsilon \in \mathbb{R}$. u and v are not connected if each edge $e \in E$ s.t. $\varpi(e) < \varepsilon$ is removed from G, but they are 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 each edge $e \in E$ has a value of core-connectivity-similarity $\varpi(e) \in [0, 1]$. T is a CCMST 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 from u to 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, the path P still remains in G. Thus, the vertices 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' from u to 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 circle 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 the 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, $\gamma_G(u, v) = \gamma_T(u, v) = \varepsilon$. □

In Fig. 4, there is a sample network. For a certain value of μ (e.g., $\mu = 4$), we first calculate the core-connectivity-similarity for each pair of adjacent vertices in the network which is indicated on the edge. Given $\varepsilon = 0.4$, if all the edges with weights less than current ε are removed from the network, four unconnected subgraphs containing two obvious dense clusters and two isolated vertices 7 and 17 will emerge. In contrast, if we first construct the CCMST of the network which is denoted by the thick lines in Fig. 4, and then partition the CCMST with the same threshold, the partitioning result on the CCMST is equal to that on the original graph. In this way, the core-connected components can be detected on the CCMST of a network.

Since each edge of the CCMST is a cut edge, different edge weights will lead to different partitioning results and form different clusters. It also means that without

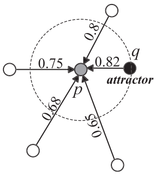


Fig. 5. An example for calculating the attachability-similarity and finding attractor.

considering the slight effect of borders all possible ε values lie in the edge weights of the CCMST. Then all the different edge weights sorted in descending order will be stored in an array W as the ε candidates.

3.3 Attractor Indices for Attaching Borders

The partition on the CCMST of a network is able to detect the cores of each cluster. In addition, there may exist some borders in a cluster. To attach the borders of each cluster efficiently, we build the indices in advance. The involved concepts and the process are given as follows.

Definition 13 (Attachability-Similarity). *Given $v \in V$, the attachability-similarity of v w.r.t. its neighbor vertices is*

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

The attachability-similarity of a vertex v is the maximum of reachability-similarities with respect to its neighbor vertices. The vertex u 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, but its attractor u is a core when $\varepsilon = AS(v)$. That is to say, v should be attached as a border to the cluster containing core u. In this case, an index of the attachability-similarity for vertex v and its attractor u will be built in advance. As shown in Fig. 5, the reachability-similarity of vertex p w.r.t. its neighbor vertices are labeled on the arrowhead lines. Since the reachability-similarity from q to p is the largest, vertex q should be the attractor of p and the attachability-similarity of vertex p is 0.82 which is equal to the reachability-similarity of p w.r.t. q. Then a record consisting of p, its attractor q and 0.82 (i.e., the attachability-similarity of p) is stored in a priority queue AIQ which is sorted in descending order of the attachability-similarity. Note that a vertex may have one or more attractors and the indices can be built during the calculation of core-connectivity-similarity. Furthermore, the procedure of attaching borders can be passed over when $\mu = 2$, because the core-connectivity-similarity of any two adjacent vertices is equal to their structure-similarity.

In this section, we have shown that the necessary information for density-based clustering of a network is stored in the CCMST and the attractor indices. In the following, we will use them to extract density-based clustering structure w.r.t. various ε values. Note that the clustering results are not sensitive to the parameter μ which can be set as a constant.

4 Clustering Extraction

In this section, we will introduce gSkeletonClu, the algorithms for extracting multilevel density-based clustering from the CCMST of a network.

core-connected maximal spanning tree

4.1 Clustering of User-Specified Parameter ε

Actually, we can easily extract the density-based clustering on the prepared CCMST for a certain value of parameter ε. The pseudocode of our algorithm gSkeletonClu for extracting clusters of the user-specified ε is given in Algorithm 1.

Algorithm 1. ExtractEpsilonClusering

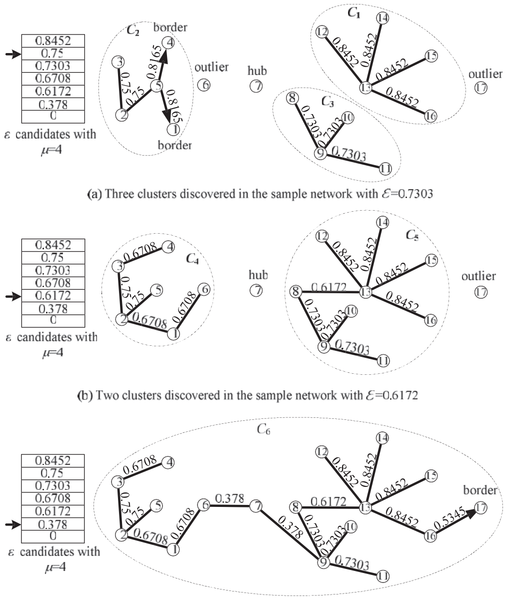
```
Input:  CCMST $T = (V, E', \varpi, W, AIQ)$ and $\varepsilon \in \mathbb{R}$
Output: A set of clusters $CR = \{C_1, C_2, \ldots, C_m\}$ and a set
        of hubs and outliers $N$
  $\tilde{E} = \{e | e \in E' \wedge \varpi(e) \geq \varepsilon\}$;
  $\tilde{T} = (V, \tilde{E})$;
  //Phase 1: Detect Core-Connected Components
  $CR = \{C[c] | c \in V, C[c]$ is a connected component in $\tilde{T}\}$;
  //Phase 2: Attach borders
  while $AIQ$.getHead().KeyValue $\geq \varepsilon$ 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 3: Detect clusters, hubs, and outliers
  $N = \emptyset$;
  for each $C \in CR$ do
      if $|C| = 1$ then
          $CR = CR - \{C\}$;
          $N = N \cup C$;
      end if
  end for
  return $CR$ and $N$;
```

Considering an initial forest of n isolated vertices, we can add the edges with the weights no less than ε to it which will group the vertices into multiple core-connected components. After attaching the borders, we will get all the clusters. For the remaining isolated vertices, they will be recognized as hubs or outliers, respectively. As shown in Fig. 6a, three clusters C_1, C_2, and C_3 as well as one hub and two outliers are detected in the sample network with $\varepsilon = 0.75$. In the cluster C_2, vertices 2, 3, and 5 are connected cores and vertices 1 and 4 are attached borders.

However, to extract density-based network clusterings with various values of parameter ε from the CCMST is not the main intended application of our algorithm. The possible extraction only demonstrates that the CCMST of a network definitely contains the information about the intrinsic density-based clustering structure. In the following, we will show that the clustering result of a proper parameter setting and the hierarchical clusters also can be extracted through an agglomerative clustering process on the CCMST.

4.2 Clustering of Automatically Selected ε

As proven above, all possible values of parameter ε can be detected on the CCMST. If a cluster validity function Q is adopted in our algorithm, the automatic selection of the parameter ε and its corresponding clustering result can be achieved conveniently. Here, the agglomerative clustering is performed on the CCMST in descending order of all the values of parameter ε. Beginning with an initial forest of n isolated vertices, we add the edges having the same



(a) Three clusters discovered in the sample network with ε=0.7303

(b) Two clusters discovered in the sample network with ε=0.6172

(c) The cluster containing all the vertices

Fig. 6. Procedure of the clustering extraction with different values of parameter ε on the CCMST of the sample network.

weights in the CCMST to the forest, from the highest value to the lowest. For each ε value, we will extract a clustering result using Algorithm 1. Then the score of the function Q of current clustering result can be calculated. The output of the algorithm is the clustering result and the correspoinding ε value which achieves the optimal value of the adopted function Q. For example, two clusters are discovered in the sample network with $\varepsilon = 0.6172$, as shown in Fig. 6b, which is the clustering result selected by our algorithm through a similarity-based modularity function.

4.3 Extraction of Hierarchical Clusters

Actually, the clustering hierarchy can be naturally revealed and pruned in the agglomerative clustering process above. The pseudocode of extracting cluster tree in a network is given in Algorithm 2.

Algorithm 2. ExtractClusterTree

```
Input: CCMST $T = (V, E', \varpi, W, AIQ)$, a threshold $\delta$
Output: A cluster tree $CT$
    $CR^{(0)} = \emptyset$; $\omega = |W|$;
    for $i = 1$ to $\omega$ do
        $\varepsilon^{(i)} = W[i]$;
        $CR^{(i)} =$ ExtractEpsilonClustering($\varepsilon^{(i)}$);
        for each cluster $C \in CR^{(i)}$ do
            for each cluster $C' \in CR^{(i-1)} \wedge C' \subseteq C$ do
                if $|C'.epsilon - C.epsilon| \geq \delta$ then
                    $C$.AddChild($C'$);
                    $C'$.Parent $\leftarrow C$;
                else
                    merge($C$, $C'$);
                end if
            end for
        end for
    end for
    $CT$.root $\leftarrow C \in CR^{(\omega)}$;
    return $CT$;
```

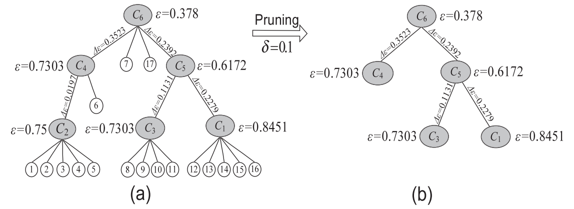


Fig. 7. Hierarchical clustering structure extracted by our algorithm on the sample network.

In the process of the agglomerative clustering, we first select the maximal weight value $W[1]$ of the CCMST and extract a clustering result CR_1 with parameter $\varepsilon = W[1]$. If we repeat the extraction above with another weight $W[2]$, we may obtain a different clustering result CR_2 with a lower value of parameter ε. It is clear that each cluster $C' \in CR_2$ consists of one or more clusters in CR_1. If $C' = C(C \in CR_1)$, then C' and C are the same cluster which will be represented by the same vertex in the hierarchical structure of the clustering. If $C' \supseteq C_1 \cup C_2 \cup \cdots \cup C_k (k \geq 2, 1 \leq i \leq k, C_i \in CR_1)$, the cluster C' should be considered as the father of the clusters C_i embedded in it. Continuing the process, the entire network will eventually be grouped into one cluster containing all the vertices in the network with the lowest value of weight on the CCMST. It should be identified as the root of the cluster hierarchy. Finally, we will obtain the whole density-based hierarchical clustering structure in the network.

As shown in Fig. 6b, two clusters C_4 and C_5 are discovered in the sample network with $\varepsilon = 0.6172$. In the next step, a cluster C_6 containing all the 17 vertices in the sample network are detected with $\varepsilon = 0.378$, as shown in Fig. 6c. Therefore, the cluster C_6 are identified as the root and clusters C_4 and C_5 are its two children. The whole cluster hierarchy of the sample network extracted by our algorithm is given in Fig. 7a.

However, there may exist some trivial information in the cluster hierarchy. For example, cluster C_2 containing vertices 1, 2, 3, 4, and 5 is detected with $\varepsilon = 0.75$. While another vertex 6 joins this cluster and they form cluster C_4 with $\varepsilon = 0.7303$. Since these two clusters are too similar in member set and size, they may be merged into one cluster so as to prune the scale and shape of the tree. In our algorithm, a threshold δ is adopted to control the process of the cluster tree pruning. If the deference of the ε values between a child cluster and its father cluster is no less than δ, we will keep the branch remaining in the tree. Otherwise, these two clusters will be merged together.

For example, in the process of extracting hierarchical clustering structure of the sample network, if we set the pruning threshold $\delta = 0.1$, clusters C_2 and C_4 will be merged. Then the reduced cluster tree of the example network is produced which becomes more meaningful and visualizable, as shown in Fig. 7b.

4.4 Analysis of Running Time Complexity

Finally, we analyze the computational complexity of our algorithm. In the stage of clustering preparation, the running time is mainly consumed in constructing the CCMST, building the attractor indices and sorting the edge weights. 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 $O(m + n\log n)$. Moreover, the edge weights can be sorted in a complexity of $O(n\log n)$. Therefore, the overall running time complexity of clustering preparation is $O(m + n\log n)$.

In the procedure of agglomerative tree clustering for extracting hierarchical clustering structure, we should deal with all the edges in the CCMST and detect connected components. It is similar to the Kruskal's algorithm for building minimal spanning tree which has a complexity of $O(m\log n)$. The running time of attaching borders has been accelerated significantly through the attractor indices which has a complexity of $O(m)$. Therefore, the running time complexity of extracting clustering tree is $O(m\log n)$. Clearly, it is also the upper bound of the complexity of extracting single ε clustering. However, the calculation of the adopted quality function Q is a little time consuming in selecting the parameter ε automatically. If its complexity is $O(t)$ and the number of edge weights is ω, the time complexity of our algorithm is $O(\omega \cdot t + m\log n)$.

5 Experiments

In this section, we evaluate the proposed algorithm using some real-world and synthetic networks. The performance of gSkeletonClu is preferentially compared with two state-of-the-art network clustering methods: SCAN [6] and CNM [13]. SCAN is an efficient density-based network clustering algorithm, and CNM is a representative modularity-based algorithm for community detection in networks. We also compare our algorithm with the algorithms ξ-cluster [9] and SCOT+HintClus [8] in extracting the density-based hierarchical clustering structure in networks. Our algorithm is implemented in ANSI C++. All the experiments were conducted on a PC with a 2.4-GHz Pentium IV processor and 4-GB RAM.

5.1 Data Sets

The performance of our algorithm is evaluated on two types of data sets. One is on the real-world networks and the other is on the computer-generated benchmark graphs with the known intrinsic clusters.

5.1.1 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 [14] and DBLP co-authorship network [15]. NCAA college-football is a social network with communities (or conferences) of American college football teams which represents the schedule of Division I-A games for year 2,000 season. The DBLP network represents the coauthor relationship between scholars in four research fields (DB, IR, DM, and ML). It is a weighted undirected network that was extracted from the DBLP computer science bibliographical data set.

TABLE 1
The Parameters of the Computer-Generated Benchmark Graphs for Performance Evaluation

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

5.1.2 Synthetic Graphs

We also use the Lancichinetti-Fortunato-Radicchi (LFR) benchmark graphs [16] to evaluate the performance of our algorithm. By varying the parameters of the graphs, we can analyze the behavior of the algorithms in detail. We generate several weighted undirected benchmark graphs with the number of vertices $n = 10{,}000$ and 100,000. The values of the parameters for the generated data sets are given in Table 1, where n is the number of vertices, m is the average number of edges, k is the average degree of the vertices, *maxk* is the maximum degree, *mu* is the mixing parameter (i.e., each vertex shares a fraction *mu* of its edges with vertices in other clusters), *minc* is the minimum cluster size, and *maxc* is the maximum cluster size. For each n, we generate 15 graphs with different mixing parameter *mu* ranging from 0.1 to 0.8 with a span of 0.05. Generally, the higher the mixing parameter of a graph is, the more difficult it is to reveal the intrinsic clusters.

Another type of synthetic network with built-in hierarchical clustering structure introduced by Arenas et al. [17] is also adopted to evaluate the performance of our algorithm in extracting hierarchical clustering structure in networks.

5.2 Cluster Validity Measures

Cluster validity checking is an important issue in cluster analysis. It uses a predefined quality function to evaluate the clustering results and guide the selection of the best one that fits the underlying data [18], [19]. In this paper, we adopt three popular cluster validity measures (i.e., Davies Bouldin index, Silhouette index and similarity-based modularity) to select the parameter ε. We consider an undirected network G, with n vertices $v_i \in V$ and m edges $e_{ij} \in E$ between vertices v_i and v_j. G is clustered into a set $\{C_1, C_2, \ldots, C_k\}$. n_i is the number of vertices and m_i the number of edges of cluster C_i. Moreover, m_{ij} is the number of edges between C_i and C_j and $m_{i\cdot}$ is the number of edges between C_i and the other clusters. Distance between v_i and v_j denoted as $d(v_i, v_j)$ is defined as the length of the shortest path between v_i and v_j in G.

Davies Bouldin index [20]. This measure is defined as

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{j\neq i}\left[\frac{diam(C_i) + diam(C_j)}{d(C_i, C_j)}\right],$$

where $diam(C) = \frac{1}{n_i^2}\sum_{u,v\in C} d(u,v)$ is the diameter of the cluster C and $d(C_i, C_j) = \frac{1}{n_{ij}}\sum_{u\in C_i, v\in C_j} d(u,v)$ is the distance of two different clusters C_i and C_j.

Silhouette index [21]. Let consider a vertex v_i that belongs to a cluster C_j. The closest cluster to vertex v_i is denoted C_h. The Silhouette index of vertex v_i is defined as

$$s(v_i) = \frac{d(v_i, C_h) - d(v_i, C_j)}{\max\{d(v_i, C_h), d(v_i, C_j)\}}.$$

For a cluster C_j, its silhouette S_j is defined as

$$s_j = \frac{1}{n_j} \sum_{i=1}^{n_j} s(v_i).$$

A global Sihouette value SI is computed by

$$SI = \frac{1}{k} \sum_{j=1}^{k} S_j.$$

Similarity-based modularity [22]. We also use the popular modularity measure Q proposed by Newman and Girvan [23] to evaluate the goodness of the clustering results. Here, a similarity-based modularity function Q_s was adopted which is extended from the connection-based modularity Q and 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 vertices within cluster C_i, $DS_i = \sum_{u \in C_i, v \in V} \sigma(u, v)$ is the total similarity between vertices in cluster C_i and any vertex in the network, and $TS = \sum_{u,v \in V} \sigma(u, v)$ is the total similarity between any two vertices in the network.

5.3 Criterion for Accuracy Evaluation

In our experiments, we adopt normalized mutual information (NMI) [24], an information-theoretic-based measurement, to evaluate the quality of clusters generated by different methods. It is currently widely used in measuring the performance of clustering algorithms. Formally, the measurement metric NMI can be defined as

$$NMI = \frac{-2 \sum_{i,j} N_{ij} log\left(\frac{N_{ij} N}{N_{i.} N_{.j}}\right)}{\sum_i N_{i.} log\left(\frac{N_{i.}}{N}\right) + \sum_j N_{.j} log\left(\frac{N_{.j}}{N}\right)}, \tag{4}$$

where N is the confusion matrix, N_{ij} is the number of vertices in both cluster 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 value of NMI ranges between 0.0 (total disagreement) and 1.0 (total agreement).

5.4 Performance Evaluation

In the following, we do a series of experiments to evaluate various performance aspects of our algorithm.

5.4.1 Extraction of User-Specified ε-Clustering

We have compared our algorithm with the density-based network clustering algorithm SCAN on several data sets. Although the clustering procedure of our algorithm is quite different from that of SCAN, the clustering results of these two algorithms are almost the same w.r.t. the same values of parameters ε and μ. But there still exists a minor difference on tackling border vertices. Since our algorithm assigns the border vertices to their attractors possessing the maximum of reachability-similarity to them. However, a border vertex is attached to the core vertex which associates with it first in the algorithm SCAN.

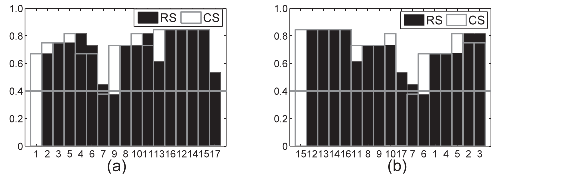


Fig. 8. The reachable-similarity plots of the sample network produced by algorithm SCOT starting from different vertices.

The density-based network clustering algorithm SCOT is extended from OPTICS. SCOT produces an ordered reachability-plot of a network by using a priority queue and a procedure similar to constructing an MST. However, we cannot consider the reachability-plot as a spanning tree of the network. Because the MST should be constructed in an undirected network with symmetric edge weights. However, the structural reachability is an asymmetric measure between pairs of neighbor vertices.

The reachability-plot of a network is determined by parameters ε_0 and μ. In extracting clusters from the ordered reachability-plot by SCOT, we should set parameter $\varepsilon \geq \varepsilon_0$ and the core-similarities of the vertices are also needed. A core v in a cluster is identified as a starting core vertex using the core-similarity of vertex v or should be associated by a neighbor core vertex u with a reachability-similarity $RS(u, v)$. Since $RS(u, v) \geq CCS(u, v)$, the clustering result w.r.t. a value of parameter $\varepsilon \geq \varepsilon_0$ extracted from the reachability-plot will yield the same result on the core vertices detected by our algorithm gSkeletonClu on the network with the same values of parameters ε and μ. Actually, SCOT can also extract the core-connected components of a network w.r.t. various values of parameter $\varepsilon \geq \varepsilon_0$ from its reachability-plot. In this respect, we may consider that the ordered reachability-plot and the CCMST of a network are equivalent for detecting cores of the density-based clusters. However, the CCMST of a network is only concerned with the parameter μ and the core-connected components w.r.t. all possible values of ε can be correctly extracted only from the CCMST of a network. Therefore, the CCMST of a network contains more integrated information for extracting core vertices.

Furthermore, some border vertices may be missed in the clusters extracted from the reachability-plot as pointed out in [9]. For example, given a border vertex v and its two neighbor vertices u and w, $RS(u, v)$ and $RS(w, v)$ may be not equal. Then, different traversing orders may lead to different reachability-similarity values of vertex v. Therefore, vertex v may belong to different clusters under the same parameter setting. To illustrate the problem above, we show the ordered reachability-plot of the sample network in Fig. 8 where the reachability-similarity is indicated by the black boxes and the core-similarity of a vertex is denoted by the white transparent box on it. The obtained traversing sequence from vertex 1 is shown in Fig. 8a. We can observe that vertices $1, 2, \ldots,$ and 7 are in the same cluster when we set $\varepsilon = 0.4$, as denoted by the horizontal white line. However, if we traverse the network from another vertex 15, we will obtain a different sequence which is shown in Fig. 8b. With the same parameter setting of $\varepsilon = 0.4$, the border vertex 7 is assigned to the cluster containing vertices

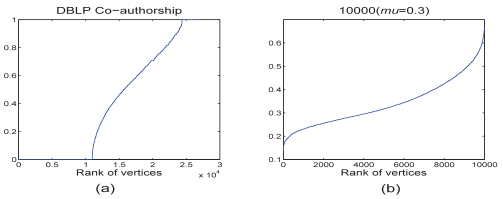


Fig. 9. Curves of sorted three-nearest structural similarity.

$8, 9, \ldots,$ and 17. Actually, a border vertex detected in a traversing sequence can also be identified as an outlier in another order of traversal. For example, vertex 17 will be recognized as an outlier if we traverse the network from it which is different from the two cases above, because it is not a core and the vertex visited after it should be selected randomly. Therefore, the clustering results on border vertices extracted by SCOT from the reachability-plot of a network depend on the traversing order of the vertices. Our algorithm gSkeletonClu does not suffer from this problem, because the border vertices are attached to their attractors which have been assigned properly in advance.

5.4.2 Automated Selection of Parameter ε

In [7] and [6], Xu et al. presented a "knee hypothesis" for the algorithms DBSCAN and SCAN, respectively, to locate the proper value of parameter ε. We sort three-nearest similarity of all vertices in the networks to locate the knees. Two curves obtained on DBLP coauthorship network and the benchmark graph 10,000 with $mu = 0.3$ are given in Fig. 9. It can be observed that there are no obvious knees 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 reachability-plot produced by SCOT, it is also difficult to select the parameter automatically, because the clustering results with various values of parameter ε have to be extracted from the plot in different cut levels, respectively.

In contrast, our algorithm can select the parameter ε automatically by using a cluster validity measure in the tree clustering process. Actually, any cluster validity measure for network clustering is able to be adopted in our algorithm. Here, we compare the performance of three widely used criteria: DBI, SI, and Q_s. For each clustering result, we can calculate the scores of various validity measures. Therefore, we can obtain the curves of the measurements w.r.t. the varying values of parameter ε in a network. For comparing the performance of different validity criteria, we also illustrate the NMI values between the clustering results for various values of parameter ε and the ground truth of the network. The obtained curves on the real-world network football and the synthetic graph 10,000 with $mu = 0.3$ are shown in Figs. 10a and 10b, respectively. In Fig. 10a, we can see that the NMI value reaches the peak when $\varepsilon = 0.522$ and it drops gradually after that. It means that the $\varepsilon = 0.522$ is the optimal parameter value since the clustering result of it is closet to the ground truth. Interestingly, the trend of Q_s scores is similar to that of the NMI values and it smoothly reaches the top at $\varepsilon = 0.522$. In Fig. 10b, we can observe that the peak values of Q_s and NMI also meet at the same clustering result with parameter $\varepsilon = 0.252$. However, the curves of DBI and SI scores fluctuate sharply and they peak obviously before and after that of the NMI, respectively.

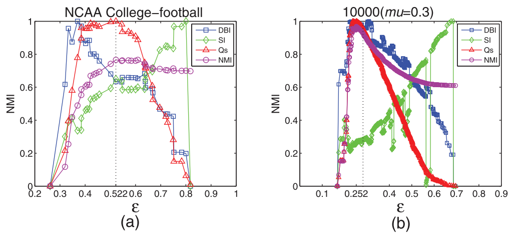


Fig. 10. The distribution of NMI values and scores of various validity measures w.r.t. clustering results of varying values of parameter ε.

The optimal ε values and the corresponding NMI values on some networks with known ground truth are given in Table 2. We try our best to select a "knee" as the ε for each network from its three-nearest similarity plot and give the NMI value of the corresponding clustering result. We also report the values of parameter ε detected by our algorithm using the validity measures Q_s, DBI, and SI, as well as the corresponding NMI values. As shown in Table 2, the manually selected ε is always not the optimal one because it depends greatly on user's intuition. The automatically selected parameter ε on football network and LFR benchmark graphs with $mu = 0.3$ are the same as the optimal ones. When the value of mixing parameter of a benchmark graph increases, though the values of parameter ε detected by Q_s is different from the optimal one, they are still very close. However, the parameter ε detected by the measures DBI and SI are quite different from the optimal ones.

It shows that the manual parameter selection method is unreliable and the Q_s measure is more effective and stable which leads to more accurate results on clustering networks.

5.4.3 Comparison of Clustering Accuracy

In the following, we compare the clustering accuracy of gSkeletonClu with SCAN and CNM on various real-world and synthetic networks. Our algorithm adopts the Q_s measure to select the parameter ε.

NCAA college-football network. This network contains 115 nodes and 613 edges. All the college football teams are divided into eleven conferences and five independent teams (Utah State, Navy, Notre Dame, Connecticut, and

TABLE 2
The Parameter ε Selected by Various Methods and the Corresponding NMI Values of the Clustering Results

Dataset	Optimal parameter ε		ε selected manually		ε selected by Q_s		ε selected by DBI		ε selected by SI	
	ε	NMI	ε	NMI	ε	NMI	ε	NMI	ε	NMI
football	0.5222	0.9414	0.5802	0.8114	0.5222	0.9414	0.3333	0.3829	0.75	0.7444
10000(mu=0.3)	0.252	0.9699	0.2223	0.6102	0.252	0.9699	0.2384	0.8033	0.4857	0.5807
100000(mu=0.4)	0.2045	0.9517	0.25	0.8585	0.1917	0.9287	0.2357	0.7595	0.3271	0.5397

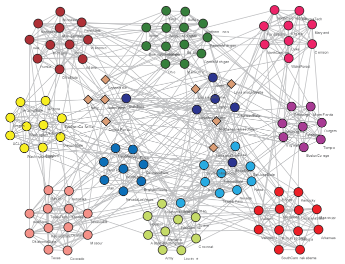


Fig. 11. Clustering result of our algorithm selected by Q_s on the NCAA college-football network.

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.

Fig. 11 illustrates the original NCAA college-football network and the clustering result of our algorithm with each vertex representing a school team. For the original network, the conferences and the group of five independent teams are indicated by cliques. Our algorithm obtains 11 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 conference are represented by the same color. Four independent teams are correctly identified as hubs. Although another four teams, which are Louisiana Monroe, Louisiana Lafayette, Louisiana Tech, and Middle Tennessee State, are identified as hubs, and other three teams are misclassified, our algorithm still performs much better than other methods including SCAN and CNM, which will be described as follow.

The SCAN algorithm detects 13 clusters as its best result in this data set with manually selected parameter setting ($\varepsilon = 0.5466$, $\mu = 3$). Two conferences are divided into two clusters, respectively. Meanwhile, it finds ten hub vertices with only four are correctly identified, and one independent team UtahState is misclassified into a conference. The modularity-based algorithm CNM discovers seven clusters, but only four clusters correctly match with the conferences. For the five independent teams, they are assigned to three different clusters. Therefore, it is obvious that our algorithm gSkeletonClu obtains better performance on this real-world network.

DBLP coauthorship network. The DBLP coauthorship network is a weighted undirected network with 28,701 vertices and 66,832 edges, in which each vertex corresponds to a distinct author and the edge between two vertices represents their coauthor relationship. The weight of an edge denotes the number of papers coauthored by these two authors.

Since the link of this real-world network is very sparse, our algorithm finds 1,441 clusters, 917 hubs, and 12,822 outliers in this network with $\mu = 3$. However, the cluster number can be reduced to 155 by setting parameter $\mu = 10$, while the automatically located parameter $\varepsilon = 0.3699$. The discovered clusters represent the real communities of academic associations much better. Due to the limited space, we can not present all the extracted communities. We then select six representative clusters and list no more than 10 cluster members along with two representative hubs or outliers in Table 3. Each community represents a group of scientists with the same research interests, such as data mining communities (32, 33), information retrieval community (56), database community (96), and machine learning community (130) in Table 3. Here, we are able to observe that our algorithm can discover meaningful co-authorship clusters. The identified hubs indicate some famous researchers who have published a large number of papers in collaboration with a variety of research groups. On the contrary, the identified outliers always correspond to those researchers who may only publish one or few papers coauthored with other scholars. Based on the results, we can see that our algorithm is also effective to identify the vertices in special roles.

The clustering results of algorithm SCAN on this network depend on the values of parameters ε and μ which is difficult to determine. The modularity-based algorithm CNM discovers 2,372 communities in this network. Among these communities, there are seven big communities including more than 500 vertices which consist of 45 percents of the vertices in the network. We observe that these big communities do not correspond to the real-world meaningful groups, because they consist of researchers from various research fields. Meanwhile there are 2,218 small communities containing no more than 10 vertices.

In summary, our algorithm gSkeletonClu can discover meaningful coauthorship clusters. It is also effective to identify vertices in special roles (i.e., hubs and outliers). We can see that the accuracy of gSkeletonClu on this large-scale network are better than that of the modularity optimization algorithm CNM, because the algorithm CNM tends to produce a small number of big communities on huge networks, due to the resolution limit problem of modularity [26].

TABLE 3
Six Communities Discovered by gSkeletonClu on DBLP Network with $\mu = 10$

community[32]	Community[33]	Community[54]	Community[56]	Community[96]	Community[130]
Philip S. Yu	Hans-Peter Kriegel	Jon M. Kleinberg	Douglas W. Oard	Jennifer Widom	Tom M. Mitchell
Jiawei Han	Martin Ester	Jure Leskovec	James Mayfield	Rajeev Motwani	Reid G. Smith
Rakesh Agrawal	Christian Böhm	Spiros Papadimitriou	Martin Franz	Jun Yang	Louis I. Steinberg
H. V. Jagadish	Peer Kröger	Jimeng Sun	G. Craig Murray	Shivnath Babu	Van E. Kelly
Divesh Srivastava	Thomas Seidl	Deepayan Chakrabarti	Jianqiang Wang	Utkarsh Srivastava	Janice S. Aikins
Qiang Yang	Jörg Sander	Eric P. Xing	Wei-Jing Zhu	Kamesh Munagala	Peter M. Will
Nick Koudas	Daniel A. Keim	Caetano Traina Jr.	Todd Ward	Arvind Arasu	S. T. Kedar-Cabelli
Jian Pei	Kai Yu	Hanghang Tong	Guangyu Zhu	Mayur Datar	Howard A. Winston
Charu C. Aggarwal	Matthias Schubert	Jia-Yu Pan	David S. Doermann	Brian Babcock	R. A. Chestek
Michael Stonebraker	Xiaowei Xu	Gueorgi Kossinets	Jian-Ming Xu	Piotr Indyk	Pat Schooley
... ...	... ...	... ...	... ...	... ...	... ...
◇Bing Liu	◇Surajit Chaudhuri	◇Christos Faloutsos	◇Dimitris Papadias	◇Surajit Chaudhuri	◇Andrew McCallum
△Bassam Bamieh	△Mathias Hampel	△Roded Sharan	△Christine D. Piatko	△Noboru Ohnishi	△Harry C. Reinstein

The last two rows are hubs and outliers associated with the corresponding communities which are labeled by ◇ and △, respectively.

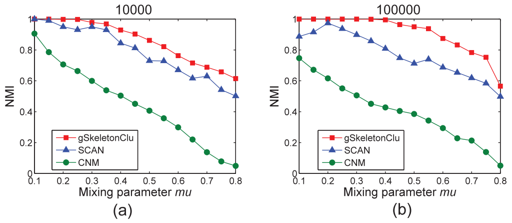


Fig. 12. NMI values of gSkeletonClu, SCAN and CNM on the computer-generated benchmark graphs.

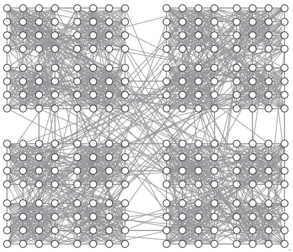


Fig. 13. The synthetic network consists of 256 vertices with hierarchical clustering structure.

LFR benchmark graphs. 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 vertex-degree and cluster-size. We compare the above three clustering algorithms on the graphs with size of 10,000 and 100,000.

The NMI scores of the three methods are plotted in Fig. 12. On most of the benchmark data sets, our algorithm gets $\mathrm{NMI} = 1$ as $mu < 0.4$, which means a perfect match with the intrinsic clustering structure. However, the performance of our algorithm decreases when $mu > 0.4$. This is because that more and more hubs and outliers are isolated in the graphs with the increasing of parameter mu. As shown in Fig. 12, the performance of gSkeletonClu is better than that of CNM on all of the generated graphs, because the algorithm CNM tends to produce small number of big clusters on the large-scale graphs.

For SCAN, the NMI values are lower than those of our algorithm but higher than CNM in most cases. This verifies the advantage of the density-based methods in cluster detection within complex networks. However, we observe that the results of SCAN are sometimes unreasonable. For example, the NMI value of SCAN on the graph with $mu = 0.6$ is higher than $mu = 0.5$ in Fig. 12b, and the NMI value of SCAN on the graph with $mu = 0.1$ is much lower than $mu = 0.2$ in Fig. 12b. This is in conflict with the characteristic of the LFR benchmark graphs which demonstrates that SCAN is sensitive to the parameter ε and the manually selected parameter cannot provide the optimal clustering results.

The NMI values demonstrate that gSkeletonClu can always locate the optimal ε and produce clustering result resembling the true clusters of the networks.

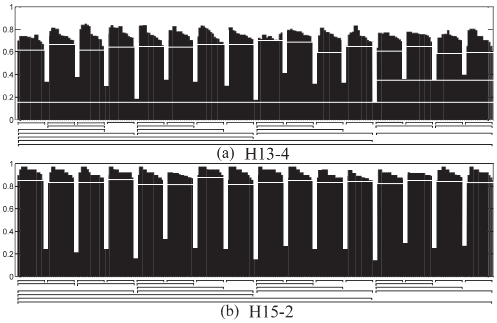
5.4.4 Extraction of Hierarchical Clustering

Finally, we evaluate the performance of our algorithm for extracting hierarchical clustering structure using synthetic networks with built-in hierarchical clusters. There are 256 vertices in the networks which are split into 16 clusters with 16 vertices on the first level, as shown in Fig. 13. The number of links of each vertex with other vertices in the same cluster at the first level is denoted by Z_{in_1}. Each four clusters are grouped into a larger cluster with 64 vertices at the second level. The links of each vertex with the vertices in other three small clusters with 16 vertices in the same cluster at the second level is denoted by Z_{in_2}. And there is only one edge of each vertex that connects with other three larger clusters in a network. The networks are indicated as H$Z_{in_1}-Z_{in_2}$, and we test the performance of our method on two widely used networks H13-4 and H15-2, respectively.

One major weakness of the algorithm SCAN is that it has difficulty to detect network clusters in varying densities and in a hierarchical structure. SCOT+HintClus has been proposed to handle this problem. Algorithm SCOT visits each vertex in a network once and produces the reachability-plot of the network. In Fig. 14, the reachability-plots of the networks H13-4 and H15-2 are shown, respectively. They are two-dimensional plots, with the ordering of the vertices on the x-axis and the reachability similarity on the y-axis. Since vertices belonging to a cluster have a high reachability similarity to their nearest neighbor, the clusters show up as mountains in the plot and they are separated by the valleys between them. The problem is that it only gives a visualized representation of the hierarchical clustering structure, which has to be analyzed by using complicated clustering extraction techniques.

In [9], a parameter ξ was introduced to extract hierarchical clusters from the distance reachability-plot of the vector data by identifying ξ-steep-up and ξ-steep-down areas in it. Actually, the method can be easily transformed to extract cluster hierarchy from the similarity reachability-plot of a network. When we set $\xi = 0.1$, the obtained cluster trees on networks H13-4 and H15-2, as the best result, are depicted underneath the x-axis in Figs. 14a and 14b, respectively. We can observe that although 16 clusters with 16 vertices and four super-clusters with 64 vertices are both correctly identified, the cluster trees both have seven levels with some middle redundant clustering results. The reason is that the steepness in the reachability-plot changes dramatically and the method is sensitive to the parameter ξ.

The algorithm HintClus extracts all possible ε-clusters from the similarity reachability-plot of a network and stores



(a) H13-4

(b) H15-2

Fig. 14. The reachability plot produced by SCOT and the hierarchical clustering structure extracted by HintClus and ξ-cluster on the synthetic networks with build-in hierarchical clustering structure.

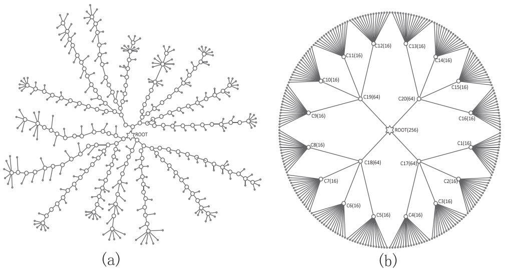


Fig. 15. The clustering hierarchy and reduced cluster tree extracted by gSkeletonClu on the network H13-4.

them in a data structure of contiguous subinterval heap. Then, it prunes the clusters whose $\Delta\varepsilon$ is within one weighted standard deviation away from the weighted mean of $\Delta\varepsilon$. The clustering results of HintClus on networks H13-4 and H15-2 are denoted by the horizontal white lines on the reachability-plots in Figs. 14a and Fig. 14b, respectively. The result on network H13-4 is relatively reasonable, where 16 clusters with 16 vertices and one super-cluster containing all 256 vertices are correctly identified. However, only one cluster with 64 vertices is discovered. The clustering result on network H15-2 contains only one level with 16 clusters of 16 vertices which is obviously over pruned. It can be seen that although this statistical pruning-based method is free from manual parameter specification, its pruning results are relatively unstable.

Different from the above two methods, our algorithm gSkeletonClu calculates the difference of the values of ε between each pair of father-child clusters along with the agglomerative clustering progress. A threshold δ is set in advance by the user and the pairs of clusters with ε difference less than δ will be merged. If $\delta = 0$, the whole cluster hierarchy will be achieved. For example, the overall hierarchical clustering of network H13-4 is given in Fig. 15a. If we set $\delta \in [0.021, 0.110]$, the standard three-level cluster hierarchy will be obtained as shown in Fig. 15b. On the network H15-2, we can obtain the same result with $\delta \in [0.011, 0.058]$.

The clustering results on the synthetic networks with build-in hierarchical clusters clearly show that our method can extract the hierarchical clustering of a network easily and can prune it flexibly by using a simple threshold parameter.

5.4.5 Running Time Comparison

To illustrate the running time of the proposed algorithm, we generate nine networks with the number of vertices n ranging from 1,000 to 500,000. For each network, the number of edges m is five times that of vertices. The running times are plotted as a function of the number of vertices in Fig. 16.

First, we compare the running time of the algorithms for extracting single ε-clustering and hierarchical clustering structure. The algorithms SCAN and SCOT+HintClus are more efficient whose performance is linear to the number of vertices and edges. It shows that the running time of our algorithm for extracting single ε-clustering and cluster tree are almost the same. They are a little slower than SCAN and SCOT+HintClus.

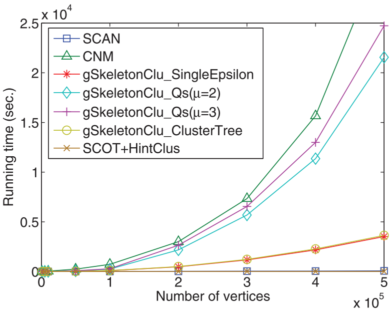


Fig. 16. The running time comparison for algorithms gSkeletonClu, CNM, SCAN, and SCOT+HintClus.

We also analyze the running time of our algorithm for locating the parameter with modularity Q_s. It shows that our algorithm is faster than CNM which has a computational complexity of $O(m \log^2 n)$. The running time of our algorithm with parameter $\mu > 2$ is a little slower than that with $\mu = 2$, because it needs a process of border attachment which has been accelerated by the prebuilt attractor indices.

In summary, though our algorithm is slower than SCAN and SCOT+HintClus, it is still quite efficient for extracting density-based clusters in networks.

6 Related Work

Clustering as an important problem in data mining has been studied for years and many clustering methods have been presented (e.g., KMeans, CLARANS [27], and so on). Especially, many real-world networks have been found possessing intrinsic communities. Since the clustering structure in networks is highly complex, many clustering methods have been introduced to solve this challenging problem, such as partitioning-based method (e.g., Metis [28] and normalized cut [29]), spectral clustering method [30], and method through modularity optimization (e.g., CNM [13]).

Density-based clustering methods. Density-based clustering approaches have been widely used in data mining owing to their ability of finding clusters of arbitrary shape and size even in the presence of noise [7], [9], [31], [32]. In 2007, Xu et al. proposed a structural network clustering algorithm SCAN [6] through extension of a density-based clustering approach DBSCAN [7]. It can find structural clusters as well as hubs and outliers in large-scale networks. However, the main difficulties of the algorithm SCAN are that it is sensitive to the parameter ε and it is hard to detect hierarchical clusters in networks. Bortner and Han proposed an algorithm SCOT+HintClus to overcome this problem through extension of the algorithm OPTICS [8]. However, it does not locate the optimal ε and needs a more sophisticated technique to extract the cluster hierarchy from the reachability-plot of a network [25]. Our work is different from the existing work and it really presents a new solution for extracting density-based clustering structure in a network.

MST-based clustering methods. The MST-based clustering method was initially proposed by Zahn [33]. The standard MST divisive algorithm removes edges from the MST in the 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 desired clusters should be given in advance and the data must have well-separable clusters in order that they can be recognized. Some clustering algorithms have been shown closely relating to MST, such as Single-link [34]. Actually, MST-based clustering algorithms have been widely used in various domains including clustering microaggregation data [35] and complex network analysis [36]. Our work introduces tree clustering into the framework of density-based clustering which is rather different from the traditional MST-based clustering method.

7 Conclusions

In this paper, we present a novel network clustering algorithm gSkeletonClu which can extract the density-based clustering structure and detect cluster hierarchy from the core-connected tree of a network. A theoretical analysis and experimental results show that the structural clustering result on the tree is equal to that on the original network. In the future, it is interesting to use our method to analyze more complex networks in various applications.

Acknowledgments

The work was supported in part by the National Science Foundation of China grants 61173093 and 61202182, the US National Science Foundation grants IIS-0905215, and the US Army Research Laboratory under cooperative agreement No. W911NF-09-2-0053 (NS-CTA).

References

[1] H. Sun, J. Huang, J. Han, H. Deng, P. Zhao, and B. Feng, "Gskeletonclu: Density-Based Network Clustering via Structure-Connected Tree Division or Agglomeration," *Proc. IEEE Int'l Conf. Data Mining (ICDM '10),* pp. 481-490, 2010.

[2] M. Girvan and M.E.J. Newman, "Community Structure in Social and Biological Networks," *Proc. Nat'l Academy of Sciences USA,* vol. 99, no. 12, pp. 7821-7826, 2002.

[3] A. Clauset, C. Moore, and M.E.J. Newman, "Hierarchical Structure and the Prediction of Missing Links in Networks," *Nature,* vol. 453, pp. 98-101, 2008.

[4] J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Proc. Symp. Discrete Algorithms (SODA '98),* pp. 668-677, 1998.

[5] P. Domingos and M. Richardson, "Mining the Network Value of Customers," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '01),* pp. 57-66, 2001.

[6] X. Xu, N. Yuruk, Z. Feng, and T. Schweiger, "SCAN: A Structural Clustering Algorithm for Networks," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '07),* pp. 824-833, 2007.

[7] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '96),* pp. 226-231, 1996.

[8] D. Bortner and J. Han, "Progressive Clustering of Networks Using Structure-Connected Order of Traversal," *Proc. 26th Int'l Conf. Data Eng. (ICDE '10),* pp. 653-656, 2010.

[9] M. Ankerst, M.M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering Points to Identify the Clustering Structure," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '99),* pp. 49-60, 1999.

[10] E.A. Leicht, P. Holme, and M.E.J. Newman, "Vertex Similarity in Networks," *Physical Rev. E,* vol. 73, no. 2, p. 026120, Feb. 2006.

[11] R. Graham and P. Hell, "On the History of the Minimum Spanning Tree Problem," *Ann. History of Computing,* vol. 7, no. 1, pp. 43-57, 1985.

[12] D.-H. Kim, J.D. Noh, and H. Jeong, "Scale-Free Trees: The Skeletons of Complex Networks," *Physical Rev. E,* vol. 70, no. 4, p. 046126, Oct. 2004.

[13] A. Clauset, M.E.J. Newman, and C. Moore, "Finding Community Structure in Very Large Networks," *Physical Rev. E,* vol. 70, no. 6, p. 066111, Dec. 2004.

[14] M. Girvan and M.E.J. Newman, "Community Structure in Social and Biological Networks," *Proc. Nat'l Academy of Sciences USA,* vol. 99, no. 2002, pp. 7821-7826, 2002.

[15] J. Huang, H. Sun, J. Han, H. Deng, Y. Sun, and Y. Liu, "SHRINK: A Structural Clustering Algorithm for Detecting Hierarchical Communities in Networks," *Proc. 19th ACM Int'l Conf. Information and Knowledge Management (CIKM '10),* pp. 219-228, 2010.

[16] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark Graphs for Testing Community Detection Algorithms," *Physical Rev. E,* vol. 78, no. 4, p. 046110, Apr. 2008.

[17] A. Arenas, A. D'iaz-Guilera, and C.J. P'erez-Vicente, "Synchronization Reveals Topological Scales in Complex Networks," *Physical Rev. Letter,* vol. 96, p. 114102, 2006.

[18] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Clustering Validity Checking Methods: Part II," *ACM SIGMOD Record,* vol. 31, no. 3, pp. 19-27, 2002.

[19] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, "Understanding of Internal Clustering Validation Measures," *Proc. IEEE Int'l Conf. Data Mining (ICDM '10),* pp. 911-916, 2010.

[20] D.L. Davis and D.W. Bouldin, "A Cluster Separation Measure," *IEEE Trans. Pattern Recognition and Machine Intelligence,* vol. 1, no. 2, pp. 224-227, Apr. 1979.

[21] P.J. Rousseeuw, "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," *J. Computational and Applied Math.,* vol. 20, no. 1, pp. 53-65, 1987.

[22] Z. Feng, X. Xu, N. Yuruk, and T.A.J. Schweiger, "A Novel Similarity-Based Modularity Function for Graph Partitioning," *Proc. Third Int'l Conf. Data Warehousing and Knowledge Discovery(DaWaK '07),* pp. 385-396, 2007.

[23] M.E.J. Newman and M. Girvan, "Finding and Evaluating Community Structure in Networks," *Physical Rev. E,* vol. 69, no. 2, p. 026113, 2004.

[24] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing Community Structure Identification," *J. Statistical Mechanics: Theory and Experiment,* vol. 2005, p. P09008, 2005.

[25] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky, "Automatic Extraction of Clusters from Hierarchical Clustering Representations," *Proc. Seventh Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining (PAKDD '03),* pp. 75-87, 2003.

[26] S. Fortunato and M. Barthélemy, "Resolution Limit in Community Detection," *Proc. Nat'l Academy of Sciences USA,* vol. 104, no. 1, pp. 36-41, 2007.

[27] R.T. Ng and J. Han, "CLARANS: A Method for Clustering Objects for Spatial Data Mining," *IEEE Trans. Knoewledge and Data Eng.,* vol. 14, no. 5, pp. 1003-1016, Sept./Oct. 2002.

[28] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM J. Scientific Computing,* vol. 20, no. 1, pp. 359-392, 1998.

[29] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.

[30] S. White and P. Smyth, "A Spectral Clustering Approach to Finding Communities in Graph," *Proc. SIAM Int'l Conf. Data Mining (SDM '05),* 2005.

[31] L. Lelis and Jörg Sander, "Semi-Supervised Density-Based Clustering," *Proc. IEEE Int'l Conf. Data Mining (ICDM '9),* pp. 842-847, 2009.

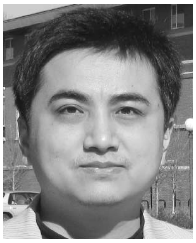
[32] X.F. Wang and D.S. Huang, "A Novel Density-Based Clustering Framework by Using a Level Set Method," *IEEE Trans. Knowledge and Data Eng.,* vol. 21, no. 11, pp. 1515-1531, Nov. 2009.

[33] C. Zahn, "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters," *IEEE Trans. Computers,* vol. CT-20, no. 1, pp. 68-86, Jan. 1971.

[34] J.C. Gower and G.J.S. Ross, "Minimum Spanning Trees and Single Linkage Cluster Analysis," *J. Royal Statistical Soc. Series C (Applied Statistics),* vol. 18, no. 1, pp. 54-64, 1969.

[35] M. Laszlo and S. Mukherjee, "Minimum Spanning Tree Partitioning Algorithm for Microaggregation," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 7, pp. 902-911, July 2005.

[36] Y. Xu, V. Olman, and D. Xu, "Minimum Spanning Trees for Gene Expression Data Clustering," *Genome Informatics*, vol. 12, p. 2001, 2001.



Jianbin Huang received the PhD degree in pattern recognition and intelligent systems from Xidian University, Shaanxi, China, in 2007. He is an associate professor in the School of Software at Xidian University. His research interests include data mining and knowledge discovery.

Heli Sun received the PhD degree in computer science from Xi'an Jiaotong University, Shaanxi, China, in 2011. She is an assistant professor in the Department of Computer Science and Technology at Xi'an Jiaotong University. Her research interests include data mining and information retrieval.

Qinbao Song is a professor at Xi'an Jiaotong University, Shaanxi, China. He has published more than 80 referred papers in the areas of data mining and software engineering. His research interests include intelligent computing, machine learning for software engineering, and trustworthy software.

Hongbo Deng received the PhD degree in computer science and engineering from the Chinese University of Hong Kong in 2009. He is a postdoctoral research associate in the Department of Computer Science at the University of Illinois at Urbana-Champaign. His research interests include information network analysis and information retrieval. He is a member of the IEEE.

Jiawei Han is an Abel Bliss professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He has been working on research into data mining and database, with more than 500 conference and journal papers. He is a fellow of the IEEE and ACM.