Combining Structured Node Content and Topology Information for Networked Graph Clustering

TING GUO and JIA WU, University of Technology, Sydney
XINGQUAN ZHU, Florida Atlantic University, Fudan University
CHENGQI ZHANG, University of Technology, Sydney

Graphs are popularly used to represent objects with shared dependency relationships. To date, all existing graph clustering algorithms consider each node as a single attribute or a set of independent attributes, without realizing that content inside each node may also have complex structures. In this article, we formulate a new networked graph clustering task where a network contains a set of inter-connected (or networked) super-nodes, each of which is a single-attribute graph. The new super-node representation is applicable to many real-world applications, such as a citation network where each node denotes a paper whose content can be described as a graph, and citation relationships between papers form a networked graph (i.e., a super-graph). Networked graph clustering aims to find similar node groups, each of which contains nodes with similar content and structure information. The main challenge is to properly calculate the similarity between super-nodes for clustering. To solve the problem, we propose to characterize node similarity by integrating *structure* and *content* information of each super-node. To measure node content similarity, we use cosine distance by considering overlapped attributes between two super-nodes. To measure structure similarity, we propose an Attributed Random Walk Kernel (ARWK) to calculate the similarity between super-nodes. Detailed node content analysis is also included to build relationships between super-nodes with shared internal structure information, so the structure similarity can be calculated in a precise way. By integrating the structure similarity and content similarity as one matrix, the spectral clustering is used to achieve networked graph clustering. Our method enjoys sound theoretical properties, including bounded similarities and better structure similarity assessment than traditional graph clustering methods. Experiments on real-world applications demonstrate that our method significantly outperforms baseline approaches.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Data Mining

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Networked graphs, super-nodes, clustering, kernel, topology

ACM Reference Format:
Ting Guo, Jia Wu, Xingquan Zhu, and Chengqi Zhang. 2017. Combining structured node content and topology information for networked graph clustering. ACM Trans. Knowl. Discov. Data 11, 3, Article 29 (March 2017), 29 pages.
DOI: http://dx.doi.org/10.1145/2996197

This research is partially sponsored by National Science Foundation under Grant No. IIP-1444949, and by the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning.
Authors' addresses: T. Guo, J. Wu (corresponding author), and C. Zhang, Faculty of Engineering & Information Technology, University of Technology Sydney, 81 Broadway, Ultimo, Sydney, NSW 2007, Australia; emails: {ting.guo-1, jia.wu}@student.uts.edu.au, chengqi.zhang@uts.edu.au; X. Zhu, Department of Computer & Electrical Engineering and Computer Science (CEECS), Florida Atlantic University, Boca Raton, FL 33431; email: xzhu3@fau.edu.

1. INTRODUCTION

Many applications, such as protein interaction networks and citation networks, use graphs to represent data entries (i.e., nodes) and their dependency relationships (i.e., edges). Based on the graph representation, the role of graph clustering [Schaeffer 2007] is to partition vertices of a large graph into different clusters based on criteria such as vertex connectivity, neighborhood similarity, and node/edge centrality scores [Newman and Girvan 2004; Xu et al. 2007; Rattigan et al. 2007].

When using graphs to represent objects, existing methods mainly use two approaches to describe node content: (1) *node as a single attribute:* Each node has only one attribute (a single-attribute node) with an edge denoting relationship between two nodes. In this article, we refer to this representation as a *single-attribute graph* (as shown in Figure 3(a)). An obvious drawback of this representation is that one single attribute cannot precisely describe the node content [Riesen and Bunke 2010]. For example, in protein interaction networks, representing each protein as a single-attribute node inherently forbids the data representation model from describing detailed information about the protein, such as protein sequence and structure information; (2) *node as a set of attributes:* An alternative solution to enhance the node content representation is to use a set of independent attributes to describe the node content (as shown in Figure 3(b)). This representation is commonly referred to as an *Attributed Graph* [Cheng et al. 2011; Xu et al. 2014; Cai et al. 1990]. In protein interaction networks, this is equivalent to representing each protein as a node with a set of attributes, such as using amino acid sequences of the protein as the node content.

Both single-attribute graphs and attributed graphs emphasize on dependency structure between objections (i.e., nodes) but overlook internal structure inside each node, where attributes/properties used to describe the node content may be subject to dependency relationships as well. For example, in a protein interaction network, each node represents one protein and edges denote protein interactions. It is known that each protein's biological function critically relies on its three-dimensional structure [Yap et al. 2013], and using sequence is insufficient to describe a protein. Alternatively, we can represent each protein as a graph, and edges between two proteins can denote their interaction relationships. As a result, a protein interaction network can be represented as a set of "*networked graphs*," where each node of the network is itself a graph.

In this article, we refer to the above "*networked graphs* " as a "*super-graph*," where each node of the network is itself a graph. A node of a super-graph is referred to as a "*super-node*." An example of this representation is shown in Figure 1.

Similarly, in a citation network, a node can represent a paper and edges denote citation relationships between papers. Using one or multiple independent key-words/attributes is insufficient to describe the content of a paper. Instead, we can represent each paper as a graph with nodes denoting keywords and edges representing contextual correlations between keywords (e.g., co-occurrence of keywords in different sentences or paragraphs). Using linked keyword relationships to form a graph representation for each paper has shown better performance than simple bag-of-words representation [Angelova and Weikum 2006]. As a result, the citation relationships between papers form networked graphs, which can be referred to as a citation super-graph with each super-node of the super-graph denoting a publication, and edges between two super-nodes denoting their citation relationships. A super-graph example is shown in Figure 2.

To find clusters from a large network, existing methods follow three common approaches: (1) Structure-based Clustering, which finds clusters based on node connectivity only; (2) Attribute-based Clustering, which uses node attribute similarity to find clusters; and (3) Structure and Attribute Clustering, which combines node attribute

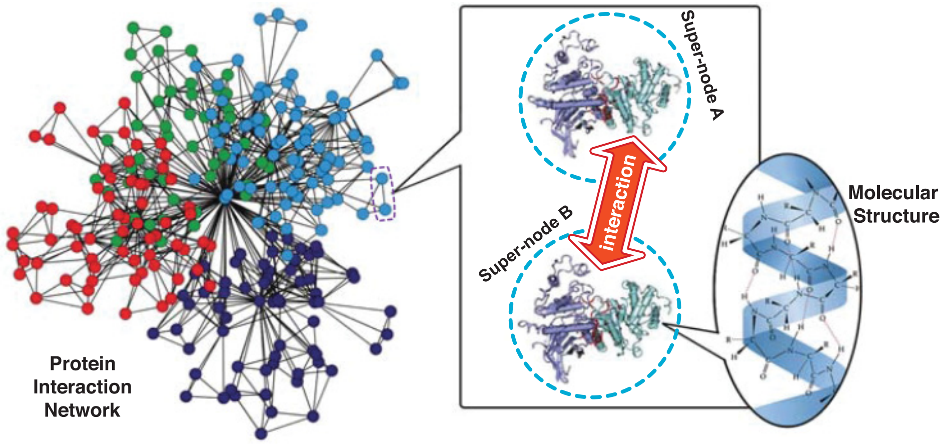


Fig. 1. A conceptual view of a protein interaction network using networked graphs to denote interaction relationships between proteins. Each protein is represented as a single-attribute graph to preserve its detailed molecular structure information. Proteins form networked graphs with edges denoting protein interaction relationships. Two interacted proteins may share functional or structure similarity [Yap et al. 2013]. In this article, the networked graphs is referred to as a "super-graph." The node of a super-graph is referred to as a "super-node," which represents a single-attribute graph (e.g., a protein).

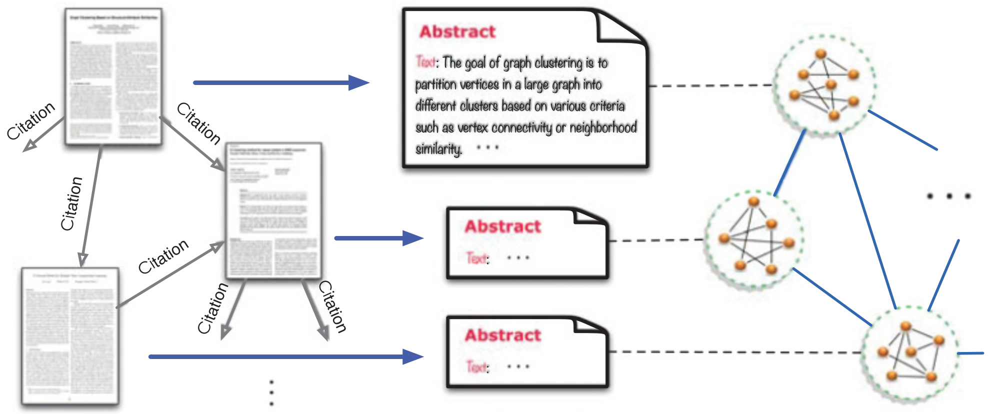

Citation
Citation
Citation
Citation
Citation
Citation
Abstract
Text: The goal of graph clustering is to partition vertices in a large graph into different clusters based on various criteria such as vertex connectivity or neighborhood similarity. · · ·
Abstract
Text: · · ·
Abstract
Text: · · ·

Fig. 2. An example of representing a scientific publication network as a super-graph (or networked graphs). Each paper (left panel) cites a number of references. For each paper, its abstract (middle panel) can be converted as a graph. So, each paper denotes a super-node, and the citation relationships between papers form a super-graph (right panel).

similarity and node connectivity to find clusters. In summary, node connectivity-based methods calculate node similarity by considering the number of paths between two nodes to keep densely connected components in clusters. Examples include normalized cut [Shi and Malik 2000], modularity [Newman and Girvan 2004], and structure density [Xu et al. 2007]. Attribute-based methods group nodes based on node content (i.e., attributes) and relationships [Tian et al. 2008; Yin et al. 2005]. Structure/attribute clustering method combines both structure and attribute to partition nodes in a large graph into groups [Cheng et al. 2011; Xu et al. 2014; Zhou et al. 2009].

All above methods are, unfortunately, inapplicable to clustering networked graphs (or super-graph) mainly because they either consider structure similarity in the network or attribute similarity between nodes, but have no solution to take internal structure inside each super-node for clustering. Indeed, super-nodes may share some *overlapped/intersected structure*, which provides useful knowledge to assess the similarity between super-nodes for clustering. In addition, the inter-connected structure between super-nodes also provides useful structure information for clustering. The structure dependency within and between super-nodes require a new clustering approach to take structure node inside the super-nodes and topology structure between super-nodes for super-graph clustering.

To apply traditional graph clustering methods to a super-graph, a simple solution is to consider the whole graph in each super-node as a single attribute (i.e., nodes are the same only if they contain exactly the same graph structure). This solution only uses the graph structure similarity without considering node content similarity, which privies important information for clustering. On the other hand, one can also consider each node in the super-node as an independent attribute, so each super-node can be regarded as a set of attributes (i.e., discarding structure information inside each super-node). After that, existing Attributed Graph Clustering methods [Zhou et al. 2009] can be applied for super-graph clustering. This solution, however, ignores the internal structure information inside each super-node.

To build clustering models for a super-graph, the main challenge is to properly calculate the similarity between super-nodes by considering attributes and internal structures inside each super-node, and inter-connectivity among super-nodes. The complex super-node structure, where each node is itself another graph, makes clustering for networked graphs a very challenging problem. More specifically,

—*Similarity between super-nodes:* Because each super-node is a graph, overlapped/intersected graph structures between two super-nodes reveal their similarity and relationships. In order to support super-graph clustering, we must properly assess similarity between super-nodes, by taking node structure and node attribute into consideration.
—*Balancing structure and content similarities:* The inter-connected structures between super-nodes and the node content both play important, yet different, roles for clustering. In an extreme case, two super-nodes in a super-graph may be linked through an edge, but do not share any overlapped attributes (i.e., no content similarity); two super-nodes may also have a significant portion of overlapped internal structures, but not have an edge in between. Properly adjusting/combining graph structure and node content similarity is crucial for achieving good clustering results.
—*Information loss and computational costs:* Traditional graph clustering methods are not suitable for super-graph representation. If we simplify each super-node as one single attribute (i.e., simplify Figure 3(c) as Figure 3(a)), the internal structure of each super-node is discarded, resulting in information loss for graph representation. We need a new design to preserve structure and node information in super-graphs for efficient super-graph clustering. In addition, for a large network, it is computationally inefficient to compare node structure similarities between every pair of super-nodes. Finding computational efficient ways to calculate similarity between super-nodes, without information loss, is a challenge.

The above challenges motivate the proposed research, which combines structured content inside each super-node and network topology between super-nodes for networked graph clustering. To minimize the computational cost, we first calculate similarity between linked super-nodes, and further extend the similarity to all super-node pairs by using matrix multiplication. For linked super-nodes, the node structure similarity is obtained by using random walk kernel and the node attribute similarity is obtained by using cosine distance. The combined similarity is used as the weight of the

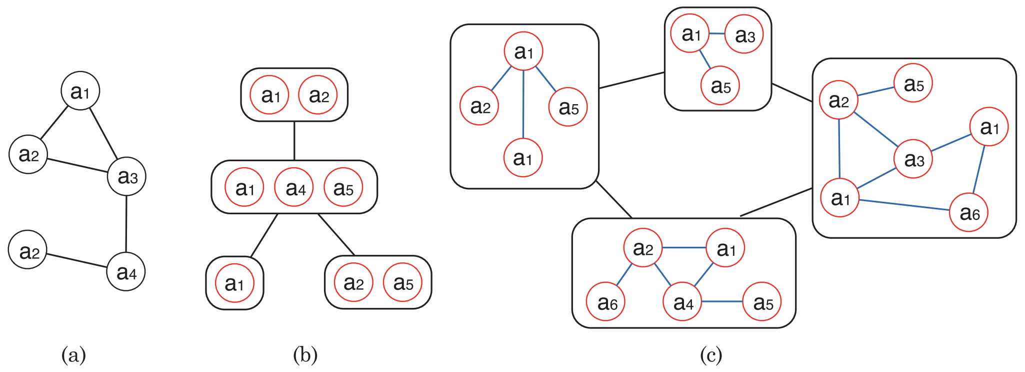


Fig. 3. Examples of a single-attribute graph (a), an attributed graph (b), and a super-graph (i.e., networked graphs) (c).

edge among linked super-nodes, so the similarity between any two super-nodes can be calculated by counting weighted random walks between them. To balance the structure and content similarities (challenge #2), some simple frequent sub-structures inside the super-nodes are used to ensure that super-nodes with shared sub-structures will have a tighter connection. The adjusted similarity matrix is fed into an iterative spectral clustering to achieve super-graph clustering, where the weights of selected sub-structures can be automatically adjusted iteratively with spectral clustering. Our method is based on a well-understood random walk theorem and enjoys sound theoretical properties, including bounded similarities combining node content and structure. Experiments on real-world data demonstrate that our method significantly outperforms baseline approaches.

The remainder of this article is structured as follows. Section 2 formulates the problem and presents the overall framework. Section 3 models the similarity between super-nodes using structure and content information. The clustering algorithm is reported in Section 4, followed by experiments in Section 5, and we conclude the article in Section 7.

2. PROBLEM DEFINITION AND OVERALL FRAMEWORK

2.1. Problem Definition

Definition 2.1 (*Single-Attribute Graph*). A single-attribute graph is represented as $g = (U, E, Att, f)$, where $U = \{u_1, u_2, \ldots, u_n\}$ is a finite set of vertices, $E \subseteq U \times U$ denotes a finite set of edges, and $f : U \rightarrow Att$ is a function from the vertex set U to the attribute set $Att = \{a_1, a_2, \ldots, a_m\}$. An attribute a_i is a single symbol, such as a keyword, to denote the node content. $\overrightarrow{Att(g)} = [a_1(g)\ a_2(g), \ldots,\ a_m(g)]$ means the attribute vector, where $a_i(g) = \sum_{j=1}^{n} h(f(u_j), a_i)$ and

$$h(f(u_j), a_i) = \begin{cases} 1, & f(u_j) = a_i \\ 0, & f(u_j) \neq a_i \end{cases}$$

In Figure 4(a), each super-node (e.g., V_1) is a single-attribute graph. An example of $\overrightarrow{Att(g)}$ for V_1 is shown in Figure 4(b).

Formally, a single-attribute graph $g = (U, E, Att, f)$ can be uniquely described by its attribute and adjacency matrix. The attribute matrix φ is defined by $\varphi_{ri} = 1 \Leftrightarrow a_i \in f(u_r)$, otherwise $\varphi_{ri} = 0$. The adjacency matrix θ is defined by $\theta_{ij} = 1 \Leftrightarrow (u_i, u_j) \in E$, otherwise $\theta_{ij} = 0$.

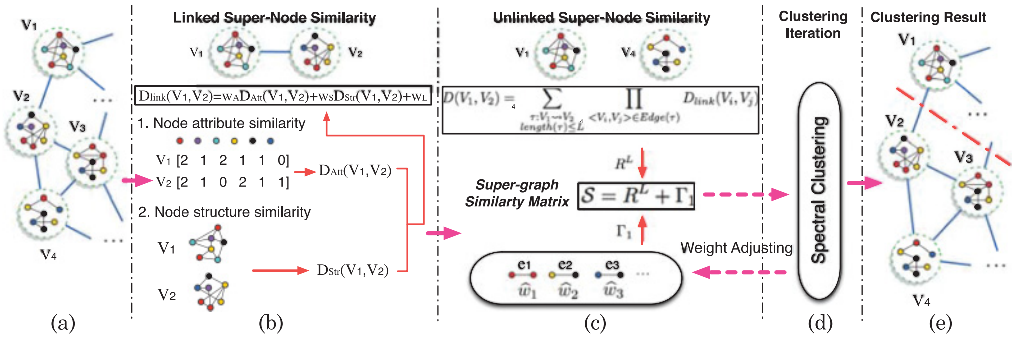

Linked Super-Node Similarity
Unlinked Super-Node Similarity
Clustering Iteration
Clustering Result
V1
V2
V3
V4
Dlink(V1,V2)=wADAtt(V1,V2)+wSDStr(V1,V2)+wL
1. Node attribute similarity
V1 [2 1 2 1 1 0]
V2 [2 1 0 2 1 1]
DAtt(V1,V2)
2. Node structure similarity
DStr(V1,V2)
Super-graph Similarty Matrix
S = R^L + Γ1
Weight Adjusting
Spectral Clustering
e1 e2 e3
(a)
(b)
(c)
(d)
(e)

Fig. 4. Networked graph clustering framework: A super-graph (a) contains a number of inter-connected super-nodes $V_1, V_2, \ldots$. To calculate similarities between super-nodes, we first find similarity between linked super-nodes by taking node attributes and internal structures into consideration (b). The similarities between unlinked super-nodes are based on weighted random walk among super-nodes, calculated using matrix multiplication (c). To balance node content and structure similarity, some weighted sub-structures (lowest row in (c)) are selected to bring super-nodes with shared sub-structure to have a tighter connection. The iterative adjustment between similarity matrix and spectral clustering (d) results in final clustering results in (e).

Definition 2.2 (*Networked Graphs*). Given a set of single-attribute graphs $\{g_1, g_2, \ldots, g_M\}$, if there exists some interconnected relationships (or edges) between some graph pairs g_i and g_j where $\{i, j\} \in [1, M]$ and $i \neq j$, we refer to $\{g_1, g_2, \ldots, g_M\}$ as networked graphs. For simplicity, we assume that the connection between two graphs (g_i and g_j) exists at the graph level but not at the node level (i.e., an edge connects g_i and g_j but does not connect particular nodes between g_i and g_j).

Definition 2.3 (*Super-Graph and Super-Node*). A super-graph is denoted by $G = (\mathcal{V}, \mathcal{E}, \mathcal{G}, \mathcal{F})$, where $\mathcal{V} = \{V_1, V_2, \ldots, V_N\}$ is a finite set of graph-structured nodes. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes a finite set of edges, and $\mathcal{F} : \mathcal{V} \to \mathcal{G}$ is an function from the vertex set $\mathcal{V}$ to $\mathcal{G}$, where $\mathcal{G} = \{g_1, g_2, \ldots, g_M\}$ is the set of single-attribute graphs. A node in the super-graph, which is a single-attribute graph, is called a *Super-node*.

In this article, *super-graph and networked graphs are equivalent terms*, so a super-graph represents a set of networked graphs. Figure 4(a) shows an example of a super-graph where $V_1, V_2, \ldots$ each denotes a super-node.

Similar to the adjacency matrix of a single-attribute graph, the adjacency matrix Θ of a super-graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{G}, \mathcal{F})$ is defined by $\Theta_{ij} = 1 \Leftrightarrow (V_i, V_j) \in \mathcal{E}$, otherwise $\Theta_{ij} = 0$. In this article, two super-nodes are called *linked super-nodes* or *linked graphs* if they are connected by one edge, or *unlinked super-nodes* otherwise.

Definition 2.4 (*Networked Graph Clustering*). Networked graph clustering (also referred to as super-graph clustering) aims to partition a super-graph G into k disjoint sub-graphs $G_i = (\mathcal{V}_i, \mathcal{E}_i, \mathcal{G}, \mathcal{F})$, where $\mathcal{V} = \bigcup_{i=1}^{k} \mathcal{V}_i$ and $\mathcal{V}_i \bigcap \mathcal{V}_j = \emptyset$ for all $i \neq j$. Ideal clustering results should achieve a good balance between the following properties: (1) super-nodes in the same cluster are close to each other in terms of their structure dependency, while super-nodes between clusters are distant from each other; and (2) super-nodes within one cluster have high similarity in both node-structure and node content, and have low similarity between super-nodes of different clusters.

2.2. Overall Framework

Figure 4 lists the major steps of our networked graph clustering framework. The essential challenge is twofold: (1) super-node similarity measurement; and (2) clustering algorithm. This is achieved through four major steps: combining node attribute and node structure similarity to find linked super-node similarity, as shown in panel (b); calculating similarity between unlinked super-node pairs in panel (c); iterative updating super-graph similarity matrix in panel (d), and final clustering in panel (e). In the next section, we first introduce the super-node similarity measurement. The clustering algorithm is reported in Section 4.

3. NETWORKED GRAPH SIMILARITY ASSESSMENT

The main challenge of networked graph clustering (or super-graph clustering) is to accurately and efficiently calculate the similarity between super-nodes. Currently, graph similarity is mainly calculated through three types of approaches: graph isomorphism [Torán 2004], sub-graph matching [Kuramochi and Karypis 2001], and graph kernel [Costa and Grave 2010; Seeland et al. 2012]. Graph isomorphism is to discover the global equivalence of two graphs and therefore can be used to assess the graph similarity. Nevertheless, graph isomorphism is NP-complete and comparing graphs via isomorphism checking is thus prohibitively expensive. Sub-graph mining intends to discover a set of frequent sub-graph patterns as features to represent each graph in as feature-vector [Guo and Zhu 2013], through which the graph similarity can be assessed. Sub-graph matching methods consider the local-structure relationship between two graphs and are robust to noise or minor changes in edges and nodes to find graph similarity. In reality, this type of methods require sub-graph pattern mining and isomorphism check as intermediate steps, and therefore can be computationally expensive. A graph kernel is a kernel function which computes the inner product between graphs, which is often used to assess the graph similarity. Graph kernels compare substructures of graphs that are computable in polynomial time and are applicable to a wide range of graphs, and have been proved to offer a faster yet principle alternative for graph similarity assessment.

Random walk [Kang et al. 2012] is one of the most popular graph kernels for measuring graph similarities. Given two graphs, a random walk graph kernel computes the number of common walks in two graphs. Two walks are common if their lengths (i.e., number of edges of the walk) are equal and the label sequences are the same (for nodes/edges labeled graphs). The computed number of common walks is used to measure the similarity of two graphs. In reality, random walk cannot be directly used to capture the similarity between super-nodes, due to the complex internal structure of each super-node. Meanwhile, given a super-graph, it is computationally inefficient to directly calculate the similarity between every pair of (linked and unlinked) super-nodes to find a similarity matrix for the super-graph. In addition, in a large super-graph G, super-nodes may have overlapped internal structures and they are also subject to linked relationships at the super-graph level. In extreme cases, two super-nodes may be linked through an edge, but do not share any overlapped attributes (i.e., no content similarity). On the other hand, two super-nodes may have a significant portion of overlapped internal structures, but not have an edge in between. Only considering super-node content or linked structures among super-nodes is inaccurate to calculate the similarity for clustering.

Accordingly, we propose a novel Attributed Random Walk Kernel (ARWK) to first calculate the similarity between linked super-nodes, and then extend the similarity to unlinked super-nodes by counting the weighted random walk among them. To balance the content and structure similarities, some frequent small sub-structures (frequent edges in the super-nodes) are selected to bring super-nodes with shared internal structures to be closer to each other in their similarity assessment.

3.1. Similarity between Linked Super-nodes

We divide the similarity between two linked super-nodes into two parts: (1) node attribute similarity, and (2) node structure similarity.

3.1.1. Node Attribute Similarity. For a super-graph, each super-node is a single-attribute graph and contains a set of attributes (Definition 2.1). Meanwhile, for super-graph clustering, we expect super-nodes in one cluster to have similar attributes, and vertices between clusters to have different attribute similarities. Accordingly, we employ cosine distance to measure the super-node attribute similarity.

Definition 3.1. *Node attribute similarity* between two super-nodes V_i and V_j is defined as follows:

$$D_{Att}(V_i, V_j) = \frac{\overrightarrow{Att(V_i)} \cdot \overrightarrow{Att(V_j)}}{||\overrightarrow{Att(V_i)}||||\overrightarrow{Att(V_j)}||} = \frac{\sum_{k=1}^{m} a_k(V_i) \times a_k(V_j)}{\sqrt{\sum_{k=1}^{m}(a_k(V_i))^2} \times \sqrt{\sum_{k=1}^{m}(a_k(V_j))^2}}, \tag{1}$$

where $Att = \{a_1, a_2, \ldots, a_m\}$ denotes the whole attribute set (which includes m unique attributes). $\overrightarrow{Att(V_i)}$ is the attribute vector of super-node V_i as defined in Definition 2.1, and an example is shown in Figure 4(b).

Because all elements in $\overrightarrow{Att(V_i)}$ are non-negative, we have $0 \leq D_{Att}(V_i, V_j) \leq 1$.

3.1.2. Node Structure Similarity. To measure structure similarity between super-nodes, we propose to use a random walk-based distance measure. Because each super-node is a single-attribute graph, we first introduce ARWK on single-attribute graphs.

Graph kernel is a kernel function that computes an inner product (i.e., a similarity function) between graphs, which is equivalent to counting shared patterns (e.g., walks, paths, subtree, or cycles) between two graphs to measure their similarity. An inherent advantage of graph kernel is that it directly measures the similarity (or distance) between two graphs without transforming graphs into a feature space, with the similarity between any two graphs being calculated by counting shared sub-structures between them.

ARWK is based on a simple idea: Given a pair of single-attribute graphs (g_1 and g_2), we compare any two nodes between them and use intersected nodes to build a new graph (i.e., a product graph) to assess their structure similarity. Because intersected nodes appear in both g_1 and g_2, we can perform random walks through these nodes on both graphs to measure similarity between g_1 and g_2 by counting number of matching walks (the walks through nodes containing the same attributes). The larger the number is, the more similar the two graphs are.

Definition 3.2 (*Single-Attribute Product Graph*). Given two single-attribute graphs $g_1 = (U_1, E_1, Att, f)$ and $g_2 = (U_2, E_2, Att, f)$, their single-attribute product graph is denoted by $g_{1\otimes 2} = (U^*, E^*, Att^*, f^*)$ ($g_\otimes$ for short), where

—$U^* = \{u | u = <u_1, u_2>, u_1 \in U_1, u_2 \in U_2\}$;
—$E^* = \{e | e = (u', v'), u' \in U^*, v' \in U^*, f^*(u') \neq \phi, f^*(v') \neq \phi, u' = <u_1, u_2>, v' = <v_1, v_2>, (u_1, v_1) \in E_1, (u_2, v_2) \in E_2\}$;

—$Att^* = Att$;
—$f^* = \{f^*(u) | f^*(u) = f(u_1) \cap f(u_2), u = <u_1, u_2>, u_1 \in U_1, u_2 \in U_2\}$.

In other words, $g_\otimes$ is a single-attribute graph whose vertex v is the intersection between a pair of nodes in g_1 and g_2. There is an edge between a pair of vertices in $g_\otimes$, if and only if an edge exists in corresponding vertices in g_1 and g_2. In the following, we show that an inherent property of the product graph is that performing a random walk on the product graph is equivalent to performing simultaneous random walks on g_1 and g_2, respectively. So, the single-attribute product graph provides an effective way to count the number of walks on nodes of both graphs without expensive graph matching.

To generate $g_\otimes$'s adjacency matrix $\theta_\otimes$ from g_1 and g_2 by using matrix operations, we define the *Attributed Product* as follows.

Definition 3.3 (*Attributed Product*). Given matrices $B \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$, and $A \in \mathbb{R}^{n'\times m'}$, the attributed product $B \boxtimes C \in \mathbb{R}^{nm\times nm}$ and the column-stacking operator $vec(A) \in \mathbb{R}^{n'm'}$ are defined as

$$B \boxtimes C = [vec(B_{*1}C_{1*})\ vec(B_{*1}C_{2*})\ \cdots\ vec(B_{*n}C_{m*})],$$

where $vec(A) = [A_{*1}^\top\ A_{*2}^\top\ \cdots\ A_{*n}^\top]^\top$, A_{*i} and A_{j*} denote i^{th} column and j^{th} row of A, respectively.

Based on Definition 3.3, the adjacency matrix $\theta_\otimes$ of the single-attribute product graph $g_\otimes$ can be directly derived from $g_1(\theta_1, \varphi_1)$ and $g_2(\theta_2, \varphi_2)$ as follows:

$$\theta_\otimes = (\theta_1 \boxtimes \theta_2) \barwedge vec(\varphi_1\varphi_2^\top), \tag{2}$$

where

$$B \barwedge \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} B_{11} \wedge x_1 & \cdots & B_{1n} \wedge x_1 \\ B_{21} \wedge x_2 & \cdots & B_{2n} \wedge x_2 \\ \vdots & \ddots & \vdots \\ B_{n1} \wedge x_n & \cdots & B_{nn} \wedge x_n \end{bmatrix}, \tag{3}$$

"$\wedge$" is a conjunction operation ($a \wedge b = 1$ *iff* $a = 1$ and $b = 1$).

As a result of the above process, we can use matrix operation to count random walk. More specifically, for the adjacency matrix θ_x of a graph g_x, each element $[\theta_x^z]_{ij}$ of this nth power matrix provides the number of length-z walks from u_i to u_j in g_x.

According to Equation (2), performing a random walk on the single-attribute product graph $g_\otimes$ is equivalent to performing simultaneous random walks on the graphs g_1 and g_2. After $g_\otimes$ is generated, ARWK, which computes the similarity between g_1 and g_2, can be defined with a sequence of weights $\delta = \delta_0, \delta_1, \ldots$ ($\delta_i \in \mathbb{R}$ and $\delta_i \geq 0$ for all $i \in \mathbb{N}$):

$$K(g_1, g_2) = \rho \sum_{i,j=1}^{n_1 n_2} \left[\sum_{z=1}^{\infty} \delta_z\ \theta_\otimes^z \right]_{ij}, \tag{4}$$

where n_1 and n_2 are the node sizes of g_1 and g_2, respectively. ρ is the control parameter used to ensure the kernel function convergence. As a result, kernel values are upper bounded (the proof is given later).

To compute the *ARWK* for single-attribute graphs, as defined in Equation (4), a diagonalization decomposition method [Gärtner et al. 2003] can be used. Because $\theta_\otimes$ is a symmetric matrix, the diagonalization decomposition of $\theta_\otimes$ exists: $\theta_\otimes = \mathcal{T}\mathcal{H}\mathcal{T}^{-1}$, where the columns of $\mathcal{T}$ are its eigenvectors, and $\mathcal{H}$ is a diagonal matrix of corresponding eigenvalues. The kernel defined in Equation (4) can then be rewritten as:

$$\begin{aligned} K(g_1, g_2) &= \rho \sum_{i,j=1}^{n_1 n_2} \left[\sum_{z=1}^{\infty} \delta_z \; (\mathcal{T}\mathcal{H}^z \mathcal{T}^{-1}) \right]_{ij} \\ &= \rho \sum_{i,j=1}^{n_1 n_2} \left[\mathcal{T} \left(\sum_{z=1}^{\infty} \delta_z \; \mathcal{H}^z \right) \; \mathcal{T}^{-1} \right]_{ij}. \end{aligned} \tag{5}$$

By setting $\delta_z = \lambda^z / z!$ in Equation (5), and use $e^x = \sum_{z=0}^{\infty} x^z / z!$, we have

$$K(g_1, g_2) = \rho \sum_{i,j=1}^{n_1 n_2} \left[\mathcal{T} \left(e^{\lambda \mathcal{H}} - I \right) \; \mathcal{T}^{-1} \right]_{ij}, \tag{6}$$

where I is an identity matrix with the same size as $\theta_{\otimes}$. The diagonalization decomposition can greatly expedite *ARWK* kernel computation.

Here, we set

$$\rho = \frac{1}{c(e^{\lambda(c-1)} - 1)}, \tag{7}$$

where

$$c = \sum_{i}^{n_1} \sum_{j}^{n_2} \left[\varphi_1 \varphi_2^{\top} \right]_{ij}. \tag{8}$$

Then, we have the following theorems.

THEOREM 3.4. *The ARWK function is non-negative definite.*

PROOF. As the random walk-based kernel is closed under products [Gärtner et al. 2003] and the *ARWK* can be written as the limit of a polynomial series with non-negative coefficients (as Equation (6)), the *ARWK* function is non-negative definite. □

THEOREM 3.5. *Given any two single-attribute graphs g_1 and g_2, the ARWK of these two graphs is bounded by* $0 \leq K(g_1, g_2) \leq 1$.

PROOF. Because $K(g_1, g_2)$ is a non-negative definite kernel by Theorem 3.4, so $K(g_1, g_2) \geq 0$. Then, we only need to show that the upper bound of $K(g_1, g_2)$ is 1.

Based on the definition of ARWK, assume the node sizes of two single-attribute graphs g_1 and g_2 are n_1 and n_2, respectively. The number of random walks on $g_1 \otimes g_2$ ($g_{\otimes}$) must be not greater than that of the complete connect graph g_c which has c nodes (as Equation (8)) because based on the definition of random walk kernel, only the nodes shared same attribute can have edges in product graph. So, the upper bound of edges in product graph is the edges in graph g_c whose size is equal to the total number of the nodes shared same attributes in g_1 and g_2. Then, we have

$$\sum_{i,j=1}^{n_1 n_2} \left[\sum_{z=1}^{\infty} \delta_z \; \theta_{\otimes}^z \right]_{ij} \leq \sum_{i,j=1}^{c} \left[\sum_{z=1}^{\infty} \delta_z \; \theta_c^z \right]_{ij},$$

where

$$\theta_c = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{bmatrix}.$$

Because

$$\sum_{i,j=1}^{c}\left[\sum_{z=1}^{\infty}\delta_z\ \theta_c^z\right]_{ij} = \sum_{z=1}^{\infty}\left\{\delta_z \sum_{i,j=1}^{c}\left[\theta_c^z\right]_{ij}\right\}$$

and

$$\sum_{i,j=1}^{c}\left[\theta_c^z\right]_{ij} = c(c-1)^z,\ \ z \geq 1,$$

then,

$$\sum_{i,j=1}^{c}\left[\sum_{z=1}^{\infty}\delta_z\ \theta_c^z\right]_{ij} = c\sum_{z=1}^{\infty}\frac{\lambda^z}{z!}(c-1)^z.$$

Because $e^x = \sum_{z=0}^{\infty} x^z/z!$, we have

$$\sum_{i,j=1}^{n_1 n_2}\left[\sum_{z=1}^{\infty}\delta_z\ \theta_{\otimes}^z\right]_{ij} \leq \sum_{i,j=1}^{c}\left[\sum_{z=1}^{\infty}\delta_z\ \theta_c^z\right]_{ij} = c(e^{\lambda(c-1)}-1).$$

So,

$$K(g_1, g_2) = \frac{\sum_{i,j=1}^{n_1 n_2}\left[\mathcal{T}(e^{\lambda\mathcal{H}}-I)\ \mathcal{T}^{-1}\right]_{ij}}{c(e^{\lambda(c-1)}-1)} \leq 1. \quad \square$$

It is worth noting that the proposed ARWK is different from the traditional random walk kernel, which simply calculates the total number of shared random walks between two graphs, but the number of shared walks can increase as the number of nodes/edges increase. As a result, the similarity based on traditional random walk kernel is unbounded. In comparison, ARWK first finds all shared nodes between two graphs (where a shared node means a node with the same attribute in both graphs), and then calculates the ratio between the number of random walks among shared nodes of two graphs and the number of random walks on the complete graph formed by using all shared nodes (the number of random walks on the complete graph is calculated by ρ^{-1} as Equation (7) which is proved in Theorem 3.5). We call this ratio the *Structure Integrity Ratio*, which is proved to be bounded. This not only provides a bounded measure for similarity assessment, but also provides an effective way to combine attribute and structure information to assess similarity between two graphs under the shared node information.

Because each super-node is a single-attribute graph, the structure similarity between two super-nodes V_1 and V_2 is equal to the ARWK of two single-attribute graphs g_1 and g_2, where $\mathcal{F}(V_1) = g_1$ and $\mathcal{F}(V_2) = g_2$:

$$\begin{aligned} D_{Str}(V_1, V_2) &= K(g_1, g_2) \\ &= \frac{\sum_{i,j=1}^{n_1 n_2}\left[\mathcal{T}(e^{\lambda\mathcal{H}}-I)\ \mathcal{T}^{-1}\right]_{ij}}{c(e^{\lambda(c-1)}-1)}. \end{aligned} \tag{9}$$

3.1.3. Linked Super-Node Similarity. Because the similarity of linked super-nodes is divided into two parts (node attribute similarity and node structure similarity), it is necessary to combine them as one similarity measure. According to Theorem 3.5, the ARWK reveals the structure similarity by measuring the structure integrity ratio of two linked super-nodes. To calculate combined similarity between super-nodes, we use three weights (w_A, w_S, and w_L) to control the degree of contributions of each part, where

w_A is the node attribute similarity weight, w_S is the weight of node structure similarity, w_L is the weight of the edge between linked super-nodes, and $w_A + w_S + w_L = 1$.

Definition 3.6 (*Linked Node Similarity*). The similarity of two linked super-nodes V_1 and V_2 in a super-graph G is

$$D_{link}(V_1, V_2) = w_A D_{Att}(V_1, V_2) + w_S D_{Str}(V_1, V_2) + w_L. \tag{10}$$

In Equation (10), w_L is a scalar variable characterizing minimum similarity between two linked super-nodes. By using w_L, if two super-nodes V_i and V_j are linked but do not share similar attributes ($D_{Att}(V_i, V_j) = 0$ and $D_{Str}(V_i, V_j) = 0$), there is still a similarity score for the linked super-node $D_{link}(V_i, V_j) = w_L$.

3.2. Similarity Between Unlinked Super-Nodes

In the above section, the similarity was calculated for each pair of linked super-nodes. For clustering purposes, we need to find similarity between all super-node pairs. Intuitively, if multiple walks connect two super-nodes V_i and V_j, it indicates that V_i and V_j have a high structure similarity. On the other hand, two super-nodes with a high structure similarity but without any edge connection between them should also have a certain degree of similarity. Motivated by this observation and the description in Section 3, we propose to use an L-length weighted random walk distance to measure the similarity between unlinked super-nodes.

Definition 3.7 (*Weighted Adjacency Matrix*). Let Θ be the $N \times N$ adjacency matrix of a super-graph G, then G's weighted adjacency matrix $\ddot{\Theta}$ is defined as follows:

$$\ddot{\Theta}_{ij} = \begin{cases} D_{link}(V_i, V_j), & V_i, V_j \text{ are linked nodes} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Definition 3.8 (*L-Length Weighted Random Walk Distance*). Assume L denotes the length limit of a random walk, the weighted random walk distance $D(V_1, V_2)$ from V_1 to V_2 is defined as

$$D(V_1, V_2) = \sum_{\substack{\tau: V_1 \rightsquigarrow V_2 \\ length(\tau) \leq L}} \prod_{<V_i, V_j> \in Edge(\tau)} D_{link}(V_i, V_j), \tag{12}$$

where τ is a walk from V_1 to V_2 whose length is $length(\tau)$ and $< V_i, V_j > \in Edge(\tau)$ means V_i and V_j are linked nodes on walk τ.

Adding weight value to the random walk is meaningful. The L-length weighted random walk distances not only consider the total number of walks between two super-nodes, but also take the similarity between linked super-nodes on each walk into account. The node similarities represent the relationship between two super-nodes in a more precise way, which, in turn, helps improve the clustering accuracy. Because each element in the weighted adjacency matrix denotes the weight of an edge appears in G, we can measure the similarity by counting the number of weighted walks and combining weights of the edges. A larger value indicates a higher similarity between two super-nodes.

As a result, the matrix form of the L-length weighted random walk distance on the whole super-graph G is

$$R^L = \sum_{i=1}^{L} \ddot{\Theta}^i. \tag{13}$$

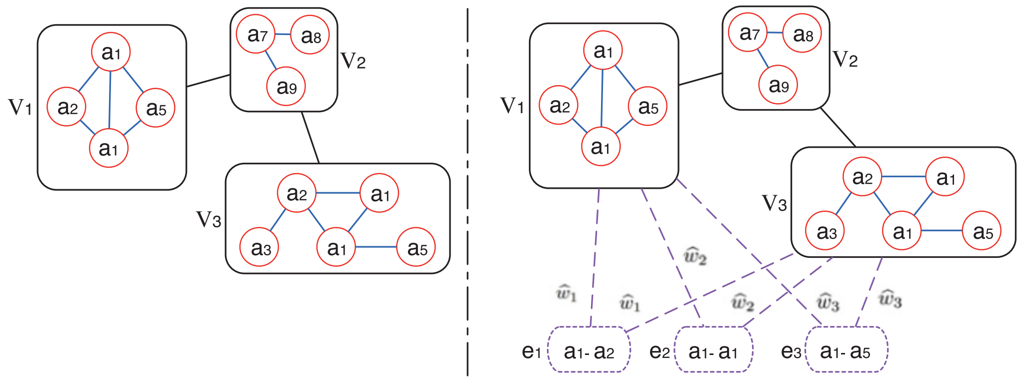


Fig. 5. An example of inaccurate similarity assessment (left panel) using weighted random walk distance. The proposed adjustment solution is shown in the right panel. The similarity between unlinked super-nodes V_1 and V_3 is denoted by $D(V_1, V_3)$. Because V_1 and V_2 have no shared attribute, and V_2 and V_3 also share no attribute, $D_{link}(V_1, V_2) = w_L$, $D_{link}(V_2, V_3) = w_L$. So, $D(V_1, V_3) = w_L^2$, which is very small. However, V_1 and V_3 indeed have similar attributes, which means they should have high attribute similarity. By using frequent sub-structures as additional nodes (right panel), the similarity between V_1 and V_3 is adjusted accordingly. In this article, we use simple edge, such as $a_1 - a_2$ as a sub-structure. $\widehat{w}_i$ is a weight on the edge.

The super-node similarity between super-node V_i and V_j (including linked and unlinked super-nodes) is

$$D(V_i, V_j) = [R^L]_{ij}.$$

3.3. Similarity Adjustment

The above similarity assessment may result in inaccurate similarity measure, as shown in Figure 5 (left panel). This is mainly because graph structure may mislead the similarity assessment. To solve this problem, we use a *frequent sub-structure relationship matrix* to build attribute–structure relationships between super-nodes. The purpose is to bring super-nodes, which do not have edge connection but share significant internal structures, closer in the similarity assessment. In Figure 5 (right panel), we visually demonstrate the solution to solve the problem as demonstrated in the left panel of Figure 5.

Because finding frequent sub-structures from graphs is a time-consuming process, we simply select top K frequent inter-edges as sub-structures from all super-nodes to generate inter-edge and super-node relationship matrix, denoted by $Q \in \mathbb{R}^{N\times K}$, where $Q_{ij} = \widehat{w}_j$ if $e_j \in E_i$, $g_i = (V_i, E_i, Att, f)$, and $Q_{ij} = 0$, otherwise. Q_{ij} means whether a super-node V_i contains inter-edge e_j or not. By using matrix Q, which explicitly captures the relationship between super-nodes and selected inter-edges, we can follow the random walk principle to assess super-node similarity with respect to the given inter-edges. More specifically, we can form a length-2 (i.e., $L = 2$) weighted random walk distance by using matrix Q as follows:

$$Z^2 = \begin{bmatrix} 0 & Q \\ Q^\top & 0 \end{bmatrix}^2 = \begin{bmatrix} \Gamma_1 & \Gamma_2 \\ \Gamma_2^\top & \Gamma_3 \end{bmatrix}. \tag{14}$$

In the above equation, Z denotes the relationship between super-nodes and selected inter-edges with different weights. Z^2 means all walks with length-2 between all super-nodes (considering each inter-edge as a special super-node). If we rearrange Z^2 as a block matrix as shown in Equation (14), $N \times N$ matrix Γ_1 means the weighted walks with length-2 between any two super-nodes. We call Γ_1 *Adding Edge Matrix*, which helps establish the relationship between super-nodes based on the inter-edges. If two super-nodes share an inter-edge e_j, there is a weight value $\widehat{w}_j^2$ between them.

In order to make the similarity matrix more accurate, we add Γ_1 to R^L as the final similarity matrix:

$$\mathcal{S} = R^L + \Gamma_1. \tag{15}$$

The matrix $\mathcal{S}$ is called the *Super-graph Similarity Matrix*, which is needed in the succeeding clustering process. The process of generating $\mathcal{S}$ is shown in Figure 4(c).

4. NETWORKED GRAPH CLUSTERING ALGORITHM

Graph clustering aims to partition nodes into densely connected sub-graphs so that nodes in the same cluster have more dense connections and higher similarities than nodes between different clusters. Accordingly, our clustering framework is to partition a super-graph G based on the similarity matrix $\mathcal{S}$ obtained in the above section. In this section, we propose an iterative spectral clustering-based super-graph clustering method to partition a super-graph G into densely connected sub-graphs, by using similarity matrix $\mathcal{S}$ obtained in the above section.

4.1. Spectral Super-Graph Clustering

In Algorithm 1, we list the proposed spectral super-graph clustering algorithm. Given a super-graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{G}, \mathcal{F})$ with N super-nodes, and the similarity matrix $\mathcal{S}$ obtained in the above process, *Laplacian matrix* $\mathcal{L}$ is defined as $\mathcal{L} = \mathcal{D} - \mathcal{S}$, where $\mathcal{D}$ is a diagonal matrix whose entries are column (or row) sums of $\mathcal{S}$. Based on this Laplacian matrix, spectral super-graph clustering aims to find k orthogonal column vectors $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ with the objective function:

$$\min_{\mathcal{X}} Tr(\mathcal{X}^\top \mathcal{D}^{-1/2} \mathcal{L} \mathcal{D}^{-1/2} \mathcal{X}) \ s.t.\ \mathcal{X}^\top \mathcal{X} = I.$$

In the above objective function, $\mathcal{X} \in \mathbb{R}^{N \times k}$ is a matrix consisting of column vectors and k is the number of clusters. These vectors are, in fact, the eigenvectors corresponding to the k smallest eigenvalues obtained from the eigen-decomposition (EVD) on $\mathcal{D}^{-1/2}\mathcal{L}\mathcal{D}^{-1/2}$. After row normalization of these eigenvectors, we can apply k-means clustering to partition these low-dimensional *embeddings* into respective clusters. For each super-node, its cluster membership is the membership of the corresponding embeddings to which the super-node belongs to.

4.2. Clustering Iteration and Weighted Self-Adjustment

Instead of relying on simple one-time clustering, it is necessary to employ an adaptive weight adjustment mechanism to iteratively assign higher/lower weights to important/unimportant inter-edges, so the clustering can converge to optimized results. This is achieved by combining clustering and weight adjustment in an iterative process.

For the frequent inter-edge relationship matrix Q of a super-graph G, we initially set all $\widehat{w}_i = 1$, $i = 1, 2, \ldots, K$, which are the weights of all K selected inter-edges. Let $\widehat{W}^t = \{\widehat{w}_1^t, \widehat{w}_2^t, \ldots, \widehat{w}_K^t\}$ be the edge weights in the tth iteration. We iteratively adjust $\widehat{w}_i^t$ with an increment $\Delta\widehat{w}_i^t$, which denotes the weight update of edge e_i between the tth and the $(t+1)$th iteration. The weight of e_i in the $(t+1)$th iteration is computed as

$$\widehat{w}_i^{t+1} = \frac{1}{2}\left(\widehat{w}_i^t + \Delta\widehat{w}_i^t\right). \tag{16}$$

The weight $\widehat{w}_i$ should be decreased. To determine whether an edge e_i is a good edge for clustering, we can use entropy measure in Equation (17), where k denotes the number

ALGORITHM 1: Networked Graph Clustering

```
Input: Super-graph $G$; Length limit $L$ of random walk paths; Parameters ($\lambda$, $w_A$, $w_S$, and $w_L$);
    Frequent inter-edge number $K$; Edge weight set $\widehat{W} = \{\widehat{w}_1, \widehat{w}_2, \dots, \widehat{w}_K\}$; Clustering iteration
    times $a$; and cluster number $k$.
Output: $k$ clusters $C_1, C_2, \dots, C_k$
 1: Initialize the weighted adjacency matrix $\ddot{\Theta}$ as an $N \times N$ zero matrix;
 2: for t = 1: a do
 3:   if $t = 1$ then
 4:     for any two linked super-nodes $V_i$ and $V_j$ in $G$ do
 5:       $\ddot{\Theta}_{ij} \leftarrow w_A D_{Att}(V_i, V_j) + w_S D_{Str}(V_i, V_j) + w_L$;
 6:     end for
 7:     $R^L \leftarrow \sum_{i=1}^{L} \ddot{\Theta}^i$;
 8:     $\widehat{w}_1^1 = \widehat{w}_2^1 = \cdots = \widehat{w}_K^1 \leftarrow 1$;
 9:     Initialize the frequent inter-edges relationship matrix $Q$ with $\widehat{W}$;
10:     Generate $\Gamma_1$ using Equation (14);
11:   else
12:     for j = 1: K do
13:       $\widehat{w}_j^t \leftarrow \frac{1}{2}(\widehat{w}_j^{t-1} + \frac{K \times H(e_j)}{\sum_{i=1}^{K} H(e_i)})$;
14:     end for
15:     Updating inter-edges relationship matrix $Q$ using changed $\widehat{W}$ and generate $\Gamma_1$;
16:   end if
17:   $\mathcal{S} \leftarrow R^L + \Gamma_1$ Update similarity matrix using Equation (15);
18:   $[C_1, \dots, C_k] \leftarrow$ Apply spectral clustering to $\mathcal{S}$ for super-graph clustering on $G$;
19: end for
```

of clusters and $P_i(j)$ means the probability of e_i in the jth cluster:

$$H(e_i) = -\sum_{j=1}^{k} P_i(j) \log_2 P_i(j). \tag{17}$$

Then, $\Delta\widehat{w}_i^t$ can be defined as

$$\Delta\widehat{w}_i^t = \frac{H(e_i)}{\frac{1}{K}\sum_{y=1}^{K} H(e_y)}. \tag{18}$$

Equation (18) ensures that the constraint $\sum_{i=1}^{K} \widehat{w}_i^{t+1} = K$ is still satisfied after weight adjustment. Then, the adjusted weight is calculated as

$$\widehat{w}_i^{t+1} = \frac{1}{2}\left(\widehat{w}_i^t + \Delta\widehat{w}_i^t\right) = \frac{1}{2}\left(\widehat{w}_i^t + \frac{K \times H(e_i)}{\sum_{y=1}^{K} H(e_y)}\right). \tag{19}$$

Using Equation (19), we can iterate clustering and edge weight adjustment several times to obtain good clustering results.

5. EXPERIMENTAL STUDY

5.1. Benchmark Data

We carry out experimental studies on five real-world networks, including Protein Interaction Network, DBLP Computer Science Bibliography network, CiteSeer scientific and academic publication network, Pubmed scientific publication networks, and Social networks for Blogger categorization (BlogCatalog). The data statistics of these networks are summarized in Table I.

Table I. Statistics of Four Benchmark Super-graphs

Super-graph	Protein	DBLP	CiteSeer	Pubmed	BlogCatalog
# super-nodes	3,390	2,019	3,312	19,717	18,617
# edges	9,946	4,222	4,732	44,338	52,376
# attributes	53	2,914	3,703	500	18,417
edge-cutting threshold	–	0.001	0.0005	0.0001	–
# edges in super-node (average)	46.77	11.45	8.27	4.23	5.21
# classes	4	2	6	3	197

Protein interaction network[1] is built based on the content of Protein Data Bank (PDB) officially released on January 1, 2013. All proteins form a network with each super-node representing a protein. For each protein, its molecular structures keep the original coordinates from PDB, and each node of protein denotes an amino acid. The edge between two proteins is built by the interaction between them. The labeling information is collected from SCOP.[2] The clustering goal is to separate proteins into four different protein superfamilies (α, β, $\alpha + \beta$, and α/β).

BlogCatalog network[3] is a rich ingredient social media dataset consisting of Bloggers, their friend circle, and blogs. Each blogger may write one or more blogs, and each blog has multiple tags and categories which could be deemed as user interests or class labels. In order to build a super-graph, we regard each blogger as a super-node, and two bloggers with established friendship are connected through an edge. For each blogger (i.e., super-node), each blog written by the blogger is represented by a set of tags. To generate a graph for each blogger, we use a complete graph to connect all tags in each blog written by the bloggers. If a blogger has multiple blogs, each blog will form a graph, and two blog graph may be connected if they share the same tags. In summary, each blogger/user corresponds to one super-node which consists of connected tags of all blogs of the user. Because each blog has one or two labels (such as video, game, education, etc., each user has multiple labels (categories). In our experiments, we calculate the accuracy on this network by checking whether the predicted label of one user belongs to the set of the user's multiple labels.

DBLP network[4] consists of bibliography data in computer science. Each record in DBLP is a scientific publication with a number of attributes such as title, abstract, authors, year, venue, and references. Our experiments build one large super-graph from DBLP by using paper content and citation relationships. More specifically, we select papers published in major Artificial Intelligence & Computer Vision conferences (AI: IJCAI, AAAI, NIPS, UAI, COLT, ACL, KR, ICML, ECML and IJCNN; CV: ICCV, CVPR, ECCV, ICPR, ICIP, ACM Multimedia and ICME) and Database and data mining conferences (DB: ICDE, SIGMOD, VLDB, EDBT, PODS, ICDT, DASFAA, SSDBM, CIKM, KDD, ICDM, SDM, PKDD, and PAKDD) to form a super-graph clustering task, where the ground truth is the research field (AI&CV or DB) that a paper (i.e., a super-node) belonging to. The clustering goal is to separate papers into groups with each group only containing papers from one single field (AI&CV or DB).

CiteSeer network[5] consists of scientific publications from six research fields, including Agents, IR (information retrieval), DB (database), AI (artificial intelligence),

[1]http://www.pdbbind.org.cn/download.asp.
[2]http://scop.mrc-lmb.cam.ac.uk/scop/parse/index.html.
[3]http://dmml.asu.edu/users/xufei/datasets.html.
[4]http://arnetminer.org/citation/.
[5]http://csxstatic.ist.psu.edu/about/data.

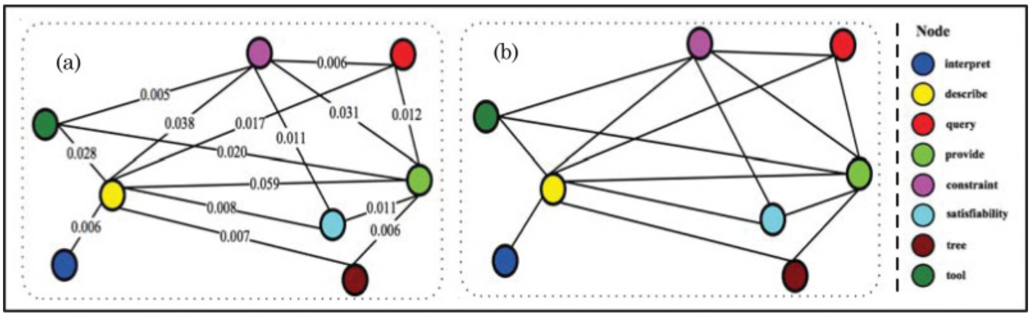


Fig. 6. A graph representation of a paper by using abstract. Each node denotes a keyword in the abstract. (a) The weight values between nodes indicate correlations between keywords. (b) By using a threshold, we can convert each abstract as an undirected unweighted graph.

HCI (human computer interaction), and ML (machine learning). The publications are collected from the CiteSeer database, which includes scientific and academic papers primarily in the fields of computer and information science.

Pubmed diabetes network[6] consists of scientific publications from the PubMed database pertaining to diabetes classified into one of three classes ("Diabetes Mellitus, Experimental," "Diabetes Mellitus Type 1," and "Diabetes Mellitus Type 2").

For DBLP, CiteSeer, and Pubmed networks, we use the abstract of each paper to denote a super-node, by constructing a fuzzy cognitive map (E-FCM) [Luo et al. 2011] that consists of keywords and their correlations. The linked keyword relationships form a graph representation for each paper (this graph representation has shown better performance than simple bag-of-words representation [Angelova and Weikum 2006]), as shown in Figure 6.

5.2. Experimental Settings

Baseline methods: Because no existing method exists for networked graph clustering, for comparison purposes, we use three baseline approaches to compare the efficiency and effectiveness of the proposed method (denoted by *SG-Cluster*).

—*SA-Cluster* is an existing graph clustering method considering both node attribute similarity and graph topological structure similarity [Zhou et al. 2009]. It uses each attribute that appeared in the graph as a new node and adds it into the graph with a changing weight value, which is adjusted during the clustering process. In the experiment, if we discard the edge information inside each super-node (i.e., each super-node contains a set of attributes), then the super-graph clustering problem is degenerated into the graph representation as shown in Figure 7(b). Then, we can use SA-Cluster for clustering (but subject to information loss inside each super-node). We choose 20 frequent attributes as new nodes and the clustering iteration time is set to 5.

—*NMF*, as a relaxation technique for clustering, has shown remarkable progress in the past decade [Lee and Seung 2001; Ding and Li 2010; Cai et al. 2011]. NMF finds a low-rank approximating matrix to the input non-negative data matrix. The most popular approximation criterion or divergence in NMF is the Least Square Error (LSE).

—*GNMF* is a graph-based approach for parts-based data representation in order to overcome the limitation that NMF fails to consider the geometric structure in the

[6]http://www.ncbi.nlm.nih.gov/pmc/tools/ftp/#Source_files.

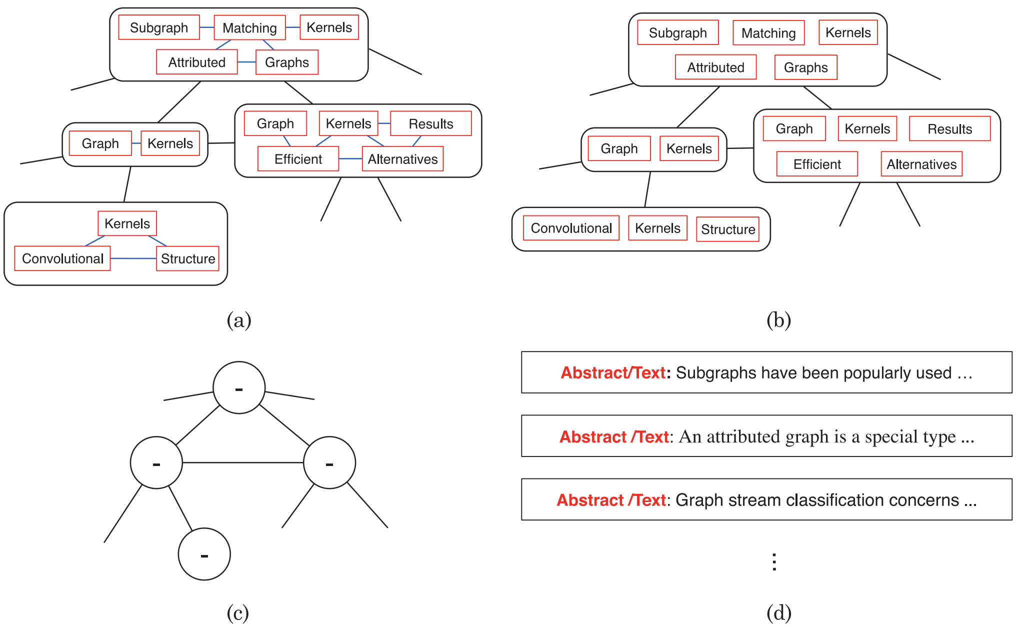

Subgraph
Matching
Kernels
Attributed
Graphs
Graph
Kernels
Graph
Kernels
Results
Efficient
Alternatives
Kernels
Convolutional
Structure
(a)
(b)
(c)
(d)
Abstract/Text: Subgraphs have been popularly used ...
Abstract /Text: An attributed graph is a special type ...
Abstract /Text: Graph stream classification concerns ...

Fig. 7. Different approaches to convert a super-graph (a) to simplified representations (so existing graph clustering methods can be applied for super-graph clustering). A super-graph (a) can be represented as an attributed graph by discarding edges in each super-node as shown in panel (b). To evaluate the impact of the super-node content to the clustering performance, we also discard all super-node contents as shown in panel (c). So, clustering is only based on topological structure of a super-graph. In panel (d), each paper is represented as one instance (using all keywords) for clustering without involving any structure or citation information.

data. It constructs an affinity graph to encode the geometrical information and seek a matrix factorization which respects the graph structure [Cai et al. 2008].

—*LP-NMTF* is a Locality Preserved Fast Non-negative Matrix Tri-Factorization approach to constrain the factor matrices of NMF to be cluster indicator matrices. As a result, the optimization problem can be decoupled, resulting in much smaller size subproblems requiring much less matrix multiplications. The authors claim that this approach works well for large-scale input data [Wang et al. 2011].

—*S-Cluster* is a baseline clustering algorithm only considering topological structures of a super-graph. We simply set the weight of each edge between two linked super-nodes to 1 (this means we do not care about the node similarity between them), and use weighted random walk distance to generate the similarity matrix for spectral clustering according to Definition 13 as shown in Figure 7(c).

—*A-Cluster* only considers super-node attribute similarity in the super-graph (i.e., using whole abstract text as one instance to represent the text/document), as shown in Figure 7(d). We use classical k-means clustering in the experiments. k-means is one of the most popular clustering methods that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, which serves as the prototype of the cluster [MacQueen 1967].

Because existing clustering methods for protein interaction networks are always based on protein sequence information. For comparison purposes, we represent content of the super-node as sequence and then apply SA-Cluster and A-Cluster accordingly.

In our experiments, the clustering results are evaluated by comparing the label of each super-node obtained from the clustering algorithms with the ground truth label

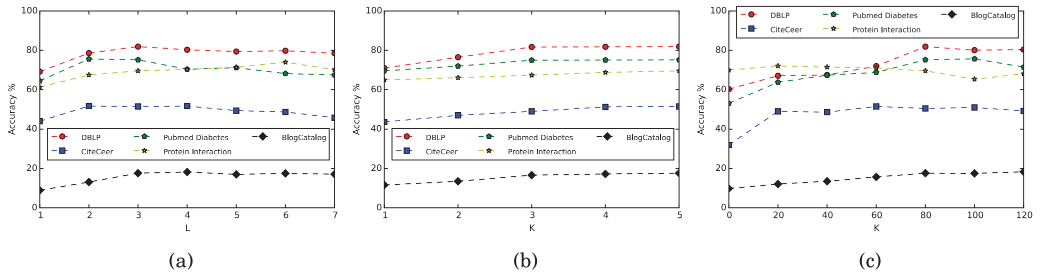


Fig. 8. Clustering accuracies on DBLP, CiterSeer, Pubmed, Protein, and BlogCatalog networks with respect to different parameter settings. (a) Accuracies w.r.t. length limit L of random walk (with 80 inter-edges and 5 clustering iteration times); (b) accuracies w.r.t. clustering iteration times (with 80 inter-edges and $L = 3$); (c) accuracies w.r.t. number of inter-edges ($L = 3$ and 5 clustering iteration times).

Table II. Clustering Results (using NMI) with Respect to Length Limit of Random Walk as Given in Figure 8

Datasets	Length limit of random walk (L)						
	1	2	3	4	5	6	7
DBLP	0.408	0.630	0.702	0.661	0.648	0.651	0.610
CiteSeer	0.189	0.302	0.310	0.322	0.271	0.261	0.243
Pubmed	0.363	0.532	0.501	0.421	0.432	0.381	0.351
Protein	0.320	0.470	0.477	0.476	0.483	0.496	0.443
BlogCatalog	0.065	0.069	0.072	0.072	0.069	0.070	0.071

provided in the network. In order to assess the performance of different algorithms, we employ two commonly used clustering performance metrics: clustering accuracy [Chen and Cai 2011] and normalized mutual information (NMI) [Strehl and Ghosh 2002]. More specifically, each super-node of our benchmark datasets (super-graphs) has a ground truth label (because they are built for classification purposes). For each super-node cluster, we will find majority class label of super-nodes in this cluster, and divide the number of super-nodes with the majority class label by the cluster size, which will result in a clustering accuracy. The total clustering accuracy is based on the average clustering accuracy across all clusters. Meanwhile, $NMI = MI(\mathcal{C}, P)/\sqrt{H(\mathcal{C})H(P)}$, where the random variables $\mathcal{C}$ and P denote the cluster and class sizes, respectively. The value of NMI is in the interval $[0, 1]$, and a larger value indicates a better clustering result. Unless specified otherwise, we default the parameter settings to $\lambda = 1, w_A = 0.4, w_S = 0.4,$ and $w_L = 0.2$.

5.3. Algorithm Performance w.r.t. Similarity Metrics and Adjustment

The proposed super-graph clustering algorithm (SG-Cluster) relies on several parameter settings: (1) the length limit of weighted random walk L, (2) the clustering iteration times, (3) the number of selected frequent inter-edges K, and (4) the weight values combining attribute and structure similarities as defined in Equation (10). Because different parameter settings result in different similarity metrics, in order to study the impact of the parameter settings to the algorithm performance, in this subsection, we vary parameter values within a range and report the algorithm performance.

5.3.1. Random Walk Length Limit, L. In Figure 8(a) and Table II, we report the performance of SG-Cluster by varying the random walk length limit (L) from 1 to 7, where Figure 8(a) reports the clustering accuracy and Table II reports the NMI values. The results show that using length-2 or length-3 walks (i.e., $L = 2$ or $L = 3$) is better than other settings for text-based data and using $L = 5$ or $L = 6$ is better for protein

Table III. Clustering Results (using NMI) with Respect to Iteration Number as Given in Figure 8

	Iteration number				
Datasets	1	2	3	4	5
DBLP	0.431	0.616	0.691	0.700	0.702
CiteSeer	0.193	0.247	0.274	0.319	0.310
Pubmed	0.405	0.440	0.527	0.532	0.501
Protein	0.355	0.401	0.449	0.462	0.477
BlogCatalog	0.060	0.064	0.071	0.072	0.072

Table IV. Clustering Results (using NMI) with Respect to Selected-Edges as Given in Figure 8

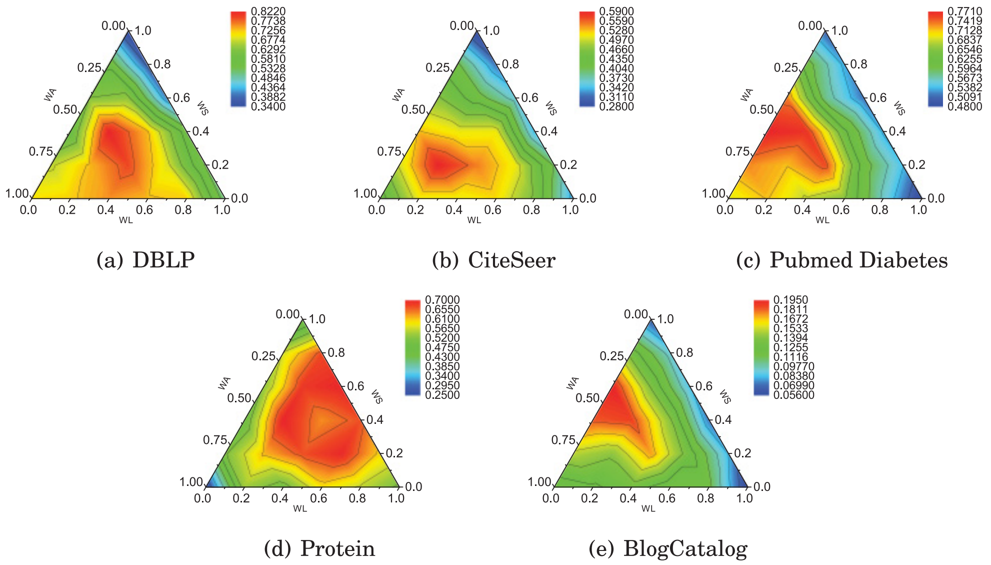
	Number of selected inter-edges (K)						
Datasets	0	20	40	60	80	100	120
DBLP	0.358	0.360	0.382	0.481	0.702	0.661	0.680
CiteSeer	0.109	0.252	0.230	0.322	0.261	0.281	0.253
Pubmed	0.333	0.342	0.411	0.421	0.462	0.531	0.501
Protein	0.503	0.522	0.471	0.432	0.419	0.401	0.451
BlogCatalog	0.055	0.058	0.061	0.065	0.072	0.070	0.071

interaction network. Longer walks ($L = 3$ or more for text-based data and $L = 6$ for protein) may deteriorate the similarity assessment. This is mainly because with a long walk length (such as $L = 7$ or higher), there are more possibilities for two super-nodes to be linked through different paths. But such paths may not reveal the genuine connection of two nodes in the network, and introduce noise to the similarity assessment. Protein interaction data need longer walks than text-based data because the sub-structures may require longer walks to traverse.

5.3.2. Clustering Convergence. In Figure 8(b) and Table III, we report the algorithm performance with respect to number of clustering iterations (Figure 8 corresponds to clustering accuracy, and Table III shows NMI values). The results demonstrate that within several clustering iterations, SG-Cluster achieves convergence on all three super-graphs, especially on DBLP and Pubmed, which only take three iterations to obtain stable clustering results. This demonstrates that SG-Cluster has a good convergence speed.

5.3.3. Frequent Inter-Edge K for Similarity Adjustment. Figure 8(c) and Table IV further report the impact of the number of inter-edges on the clustering accuracy (Figure 8(c)) and NMI values (Table II). For DBLP and Pubmed, the accuracy increases with an increasing number of inter-edges. For CiteSeer and BlogCatalog, SG-Cluster shows relatively stable results with different inter-edge numbers. This demonstrates the number of inter-edges are not vital for SG-Cluster, partially because the weight adjustment mechanism will adaptively fine tune the contribution of each inter-edge, depending on the actual number of inter-edges involved in the clustering process. For protein interaction network, the trend is different. This is mainly because the attribute set for protein is very small (as shown in table I). An edge may always appear repeatedly in different types of proteins. So, one single edge may not contain discriminative information for clustering.

5.3.4. Attribute vs. Structure Similarity. In Figure 9, we exploit the relationships among different similarity metrics (i.e., node attribute similarity, node structure similarity, and link information) in our SG-Cluster method through adjusting weight parameters w_L, w_S, and w_A as defined in Equation (10). Under the constraint of $w_A + w_S + w_L = 1$, ternary contours show the clustering accuracy changes with respect to different weight



(a) DBLP (b) CiteSeer (c) Pubmed Diabetes

(d) Protein (e) BlogCatalog

Fig. 9. The ternary contours of clustering accuracies w.r.t. different weight parameters w_L, w_S, and w_A as defined in Equation (10). The clustering accuracy values are color coded as shown in the right panels, respectively.

parameter settings. If w_S and w_L were both set as 0, and $w_A = 1$, this would lead to a simple content similarity metrics-based spectral clustering.

The results show that each of the three parts plays unique and important roles for clustering on the five networks. Meanwhile, Figure 9 also shows that simply relying on either structure or node attribute similarity, respectively (i.e., results corresponding to the corners of each triangle), without similarity adjustment, often result in deteriorated clustering performance. This is understandable because structure and node content each has its own contribution for similarity assessment in different domains. For protein super family data, the node structure and link between nodes are more important than attribute similarity, because protein attribute sets, which corresponds to amino acids, are almost the same in different protein superfamilies. On the other hand, for citation network, the node attribute and link information are most useful, because keywords and related citations better illustrate the topics of articles. Therefore, for real-world applications, the adjustment strategies of weight parameters highly depend on the data domain themselves.

5.4. Performance Comparisons Between Different Methods

In order to validate the impact of each respective factor, including super-graph structure, super-node structure, and super-node attribute, on the clustering performance, we apply different baseline methods to the super-graphs. During this process, super-graphs are converted into different representations, in order to suit for the input requirement of each baseline method. In Figure 10 and Table V, we report the clustering results by using accuracy as the performance measure.

The text-based clustering results in Figure 10 and Table V show that super-graph structures (S-Cluster) provide very limited knowledge for clustering. This is mainly because S-Cluster does not consider structured content information inside each super-node but only relies on super-graph topological structures for clustering. Comparing with node content and topological structures, we find that node content plays a more

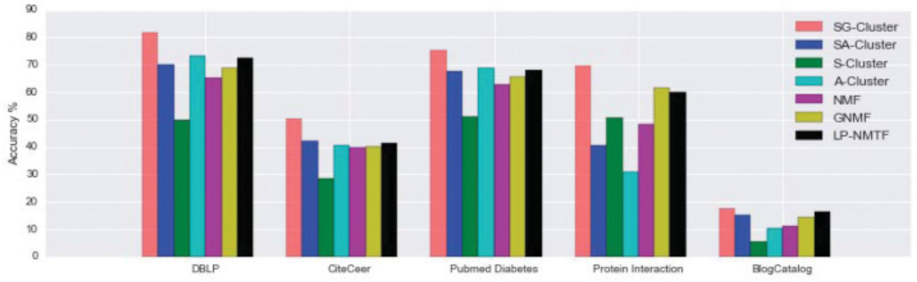


Fig. 10. Clustering accuracies of different clustering methods (SG-Cluster, SA-Cluster, S-Cluster, A-Cluster, NMF, GNMF, and LP-NMTF) on five benchmark super-graphs (DBLP, CiterSeer, Pubmed, Protein, and BlogCatalog).

Table V. Clustering Results (using NMI) with Respect to Different Clustering Methods as Given in Figure 10

	NMI						
Datasets	SG-Cluster	SA-Cluster	S-Cluster	A-Cluster	NMF	GNMF	LP-NMTF
DBLP	**0.702**	0.413	0.281	0.631	0.407	0.410	0.534
CiteSeer	**0.412**	0.287	0.071	0.202	0.214	0.231	0.233
Pubmed	**0.600**	0.382	0.317	0.402	0.359	0.376	0.393
Protein	**0.581**	0.306	0.384	0.326	0.362	0.537	0.501
BlogCatalog	**0.072**	0.063	0.021	0.035	0.042	0.051	0.069

The best performance on each dataset is bold-faced.

important role in node similarities. Only on data with strong structural dependency, such as Protein network, S-Cluster can achieve good performances.

SA-Cluster considers both node content (by converting each super-node as a set of attributes) and super-graph structure information for clustering, but it discards structure information inside each super-node (so each super-node contains a set of independent attributes). This has shown to be ineffective for super-graphs whose nodes also contain rich structured information. By using ARWK to compare super-node content, and integrating inter-edges to balance the node content and structure similarity, SG-Cluster demonstrates much better performance than SA-Cluster.

Interestingly, by using text information of each paper for clustering, the results in Figure 10 show that A-Cluster obtains surprisingly good results. This is mainly because A-Cluster uses all keywords in the abstract for clustering, whereas other three methods employ a graph representation where the selected threshold will remove some keywords from the abstract and result in a much smaller node space (compared to the original keyword space). Despite noticeable information loss in the graph converting process, SG-Cluster obtains the best performance on three super-graphs. This is mainly because SG-Cluster combines super-graph topological structure, super-node structure, and super-node attribute information for node similarity assessment, which essentially helps improve the clustering quality.

NMF/NMTF-based co-clustering methods (NMF, GNMF, and LP-NMTF) try to build instance–instance nearest neighbors graph and enforce the k-NN graph in the objective function to discover cluster structures with respect to low dimensional feature space (i.e., manifold). However, similar to SA-clustering method, they do not take structure information inside super-nodes into consideration.

The results from protein clustering show that sequence is not a good representation for protein superfamily clustering. It is well known that protein interaction relationships (edges) provide useful information for protein categorization. Our results show that SG-Cluster achieves better performance than other three methods because it uses

Table VI. Runtime Performance Comparisons

Clustering methods	DBLP	CiteSeer	Protein interaction	BlogCatalog
SG-Cluster	5694.27s	5979.32s	9327.65s	25145.55s
SA-Cluster	1765.57s	1931.65s	2364.90s	5263.17s
S-Cluster	72.86s	84.34s	93.44s	361.34s
A-Cluster	776.95s	821.67s	1357.27s	1254.43s
NMF	1644.23s	1941.77s	2367.75s	3963.81s
GNMF	5712.19s	6024.28s	10527.79s	28342.75s
LP-NMTF	4100.95s	3971.54s	6810.02s	9216.43s

Table VII. One Cluster by using SG-Cluster Method with $k = 100$

ID	Paper title	Conference	Ground truth class
502386	Hierarchical filtering method for content-based music retrieval via acoustic input	ACM Multimedia	AI&CV
600022	*Warping indexes with envelope transforms for query by humming*	*SIGMOD*	*DB*
502465	A practical query-by-humming system for a large music database	ACM Multimedia	AI&CV
503040	Manipulation of music for melody matching	ACM Multimedia	AI&CV
502387	Super MBox: an efficient/effective content-based music retrieval system	ACM Multimedia	AI&CV
503227	Music scale modeling for melody matching	ACM Multimedia	AI&CV
503632	Scalable music recommendation by search	ACM Multimedia	AI&CV
1250100	Query expansion for hash-based image object retrieval	ACM Multimedia	AI&CV

molecular structure as protein representation and takes both the internal and external structures of super-nodes into consideration.

5.5. Runtime Comparison and Analysis

In Table VI, we report the runtime performance of different methods. The results show that S-Cluster is the most efficient method, mainly because S-Cluster does not consider node content similarity (between super-nodes) for clustering. A-Cluster needs to transfer each text into feature space and only carries out the clustering once. SA-Cluster requires a significant amount of time on the added new nodes, including establishing relationships between new nodes and original nodes, and the iteration for weight adjustment, which add significant runtime overhead. SG-Cluster is mostly time consuming comparing to other methods. This is mainly because SG-Cluster intends to achieve accuracy gain by using kernel between linked super-nodes, which is time consuming. As shown in Figure 10, the runtime overhead of SG-Cluster is paid off through the clustering quality improvement. The runtime on Protein network is much longer than other networks because protein super-nodes have a much bigger size on average.

5.6. DBLP Case Study

To further study the detailed cluster results, we examine some clusters on DBLP super-graph by setting the number of clusters as $k = 100$. We report detailed cluster members of one cluster in Table VII. The cluster has eight members (i.e., papers), and most of the papers in this cluster are from the same class ($AI\&CV$), as well as the same conference

Table VIII. The Linked Papers of Paper 600022

ID	Paper name	Conference	Belonging class
502386	Hierarchical filtering method for content-based music retrieval via acoustic input	ACM Multimedia	AI&CV
502465	A practical query-by-humming system for a large music database	ACM Multimedia	AI&CV
599512	Similarity-based queries for time series data	SIGMOD	DB
503040	Manipulation of music for melody matching	ACM Multimedia	AI&CV
599619	Optimal multi-step k-nearest neighbor search	SIGMOD	DB
642592	Scaling and time warping in time series querying	VLDB	DB

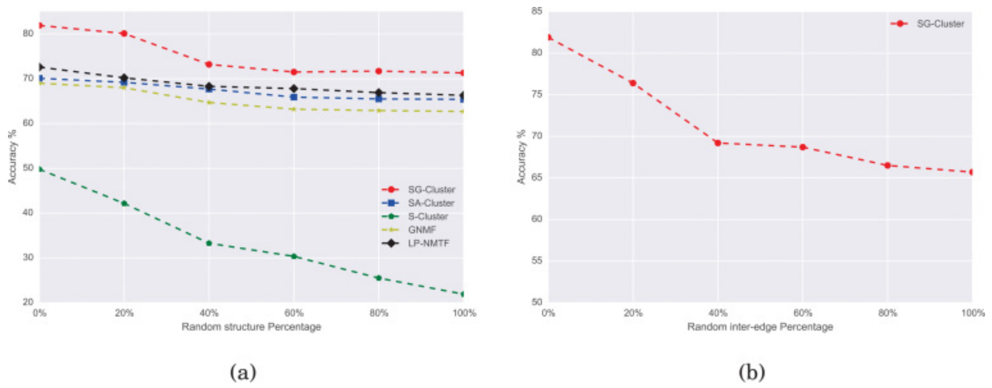


Fig. 11. Clustering accuracies on DBLP with different structure/inter-edge settings for different method. (a) Accuracies w.r.t. random structure percentages (replacing edges with a given percentage in super-graph randomly) and (b) accuracies w.r.t. random inter-edge percentages (replacing inter-edges with a given percentage in super-nodes randomly).

ACM Multimedia. From the titles of the papers, it is clear that paper topics are highly correlated to each other (most of them are related to the *Music retrieval*). It means that the proposed SG-Cluster method can merge super-nodes (i.e., papers) with similar content into respective clusters. There is one exception (ID = 600022 as highlighted in italic format). The paper actually belongs to another class (*DB*) and different conference (*SIGMOD*). We further checked the citation information as shown in Table VIII. The linked papers of this article belong to different conferences (three *AI&CV* papers and three *DB* papers). It means Paper 600022 is close to different topics. Its references or cited papers belong to different topics. So, the paper is itself cross multiple fields. This maybe the main reason resulting in the wrong clustering result.

In Figure 11, we report our case study results in investigating the importance of super-graph's topology structures vs. super-node's internal structures. To study super-graph's topology structure, we first randomly remove a given percentage of the edges (which connect super-nodes) from the original super-graph. After that we randomly generate edges between super-nodes in the super-graph with the number of added edges equal to the number of removed edges, and the results are reported in Figure 11(a). Our purpose is to randomly change the super-graph topology structures (but keep the edge density remain the unchanged) and observe the algorithm performances. Because not all baseline methods take topology structure of super-graph into consideration, in Figure 11(a), we only report the changing clustering results by using SG-Cluster, SA-Cluster, S-Cluster, GNMF, and LP-NMTF.

In addition, because our proposed method (SG-Cluster) also takes super-node's internal structure into consideration for clustering, compared to other baseline methods, we also report the changing clustering results by replacing edges inside super-nodes by using random edges. Similar to super-graph's topology structure study, we also randomly select a number of edges from super-nodes and using random edges to replace the same number of selected edges. The results are reported in Figure 11(b).

From this case study, we find that structures inside super-nodes place a more important role for clustering task comparing with super-graph's structure. In the proposed SG-Cluster, both network topology structures and internal structures are utilized to achieve better accuracies comparing to other baseline methods.

6. RELATED WORK

While no work currently exists for networked graph clustering, our research is related to clustering, clustering small graphs, and clustering large networks.

6.1. Clustering

Clustering is an established research topic in data mining. Its applications are commonly observed in many domains such as computer vision, biology, etc. In traditional clustering, the aim is to divide an unlabeled data set into groups of similar data points. This can be achieved by comparing feature-based similarities/distances between instance pairs, and assigning each instance to the group to which it is mostly similar. *k*-means [Bishop 2006] is the classical clustering method which follows the traditional clustering principle. From a geometrical point of view, a data set can be seen as a set of nodes connected with structure relationships, and clustering aims at finding intrinsic groups of the data. Common approaches for clustering can be summarized as follows.

Spectral clustering methods were introduced to the machine learning community as elegant solutions to solve partition problems, where the objective is to make a graph cut (a bisection of the graph). This is usually done by (1) defining a graph Laplacian with a graph cut objective in mind [von Luxburg 2007], (2) finding the significant eigenvector of the Laplacian (e.g., the second smallest eigenvector for the normalized cut objective) [Smalter et al. 2010], and (3) thresholding the eigenvector. Nodes corresponding to elements of the eigenvector above the threshold belong to one partition, and those below belong to the other [Ng et al. 2001].

Hierarchical clustering is a defined as a hierarchical structure, where each top-level cluster is composed of sub-clusters and so forth. This is useful in situations where the graph structure itself is hierarchical, and a single cluster can naturally be composed further to obtain a more fine-grained clustering or alternatively merged with another cluster to obtain a coarser division into clusters. Clustering methods that produce multi-level clusterings are called hierarchical clustering algorithms and can be further divided into two classes, depending on whether the partition is refined or coarsened during each iteration: (1) top-down or divisive algorithms that split the dataset iteratively or recursively into smaller and smaller clusters [Brandes et al. 2003; Bui et al. 1987; Flake et al. 2004; Fortunato et al. 2004; Girvan and Newman 2002], and (2) bottom-up or agglomerative algorithms that start with each data element in its own singleton cluster or another set of small initial clusters, iteratively merging these clusters into larger ones [Carrasco et al. 2003; Donetti and Muñoz 2004].

Non-negative matrix factorization (NMF) [Saigo et al. 2009] are typical methods which carry out clustering from the geometrical point of view. Some researches have also been proposed to combine traditional clustering and geometrical relationships between instances for better clustering results (commonly referred to as Attributed Graph Clustering [Cheng et al. 2011; Xu et al. 2014; Zhou et al. 2009]).

6.2. Clustering Small Graphs

Small graph clustering aims to cluster a number of (small) graphs into different groups, with each group containing graphs sharing similar structure information [Seeland et al. 2011, 2012; Aggarwal et al. 2010]. This type of applications are typical for proteins, chemical compounds, and many other domains where each object can be represented as a graph, and the clustering goal is to find objects sharing similar structure properties/characteristics [Le et al. 2004]. For example, by representing each computer software as a graph, graph clustering has been used to identify common malware behaviors in order to detect malicious software [Park et al. 2013]. Graph clustering has also been used to trace and capture social data evolution [Giatsoglou and Vakali 2013].

For most methods in this category, the key challenge is to properly assess the similarity between two graphs, such that similar graphs can be merged into groups. To achieve the goal, two types of approaches are commonly used, including (1) substructure-based methods, and (2) graph kernel-based approaches. For substructure-based methods [Seeland et al. 2011; Aggarwal et al. 2010; Park et al. 2013; Le et al. 2004], the objective is to find a set of substructures (such as sub-graphs, paths [Aggarwal et al. 2010], or cliques [Chi et al. 2013]) to convert graph into feature vector space to calculate similarities between graphs [Guo and Zhu 2013]. Alternatively, graph kernel-based methods [Costa and Grave 2010; Seeland et al. 2012] directly use kernels to find similarity between graphs without explicitly converting graphs into vector space.

Our networked graph clustering is different from small graph clustering mainly because that existing works regard each graph as being independent without considering their structure relationships. Instead, we allow graphs to share dependency relationships, and will take such relationships into consideration for graph clustering.

6.3. Clustering Large Networks

Our research is mostly related to clustering large networks, where existing work main focus on grouping vertices of a network into clusters, by taking topology structures of the network into consideration. The expected clustering results typically require that edges in each cluster are relatively more frequent than edges across the clusters. This problem is also referred to as graph clustering in the literature [Schaeffer 2007; Dhillon et al. 2005; Chen et al. 2014]. In biomedical research, clustering node sharing dense connections has been used to determine colon cancer pathway [Zhu et al. 2013] and find genes with similar patterns by using their interactions [Amir et al. 1999]. In social network analysis, finding cluster of nodes with similar characteristics can serve as building blocks for many other tasks, such as finding communities or other meaningful structures from the networks [Fortunato 2010; Newman and Girvan 2004; Danon et al. 2005]. Xu et al. proposed a structure clustering algorithm named SCAN, which clusters vertices based on a structure similarity measure [Xu et al. 2007]. A recent survey [Malliaros and Vazirgiannis 2013] summarizes the research on clustering and community detection in directed networks. For most node clustering-based community detection methods, the similarity measures are mainly based on topology of the network. In a recent research [Yang et al. 2013], the authors have proposed to combine edge structure and node attributes for community detection (but they only consider that each node contains a set of attributes, i.e., an attributed-graph [Cheng et al. 2011]).

Our networked graph clustering is different from existing works in this category mainly because we allow each node in the network to be represented as a graph. This relaxation makes our model effective for capturing and representing objects containing structure information.

7. CONCLUSIONS

This article formulates a new networked graph clustering problem, where graphs are subject to inter-connected relationships and the clustering goal is to find cluster of graphs sharing similar content and structures. A noticeable feature of networked graphs (also referred to as a super-graph in this article) is that we allow each object to be represented as a graph, in order to effectively preserve the content and structure information of the object. In addition, the inter-connected relationships between graphs also allow our model to capture inter-relationships between objects, such as interactions between proteins or citation relationships between publications.

Due to the complex structure representation in networked graphs, we proposed to calculate the similarity between two graphs by assessing (a) node attribute similarity, and (b) node structure similarity. In addition, some sub-structures are used to balance the node structure and node content similarities, with graphs sharing similar internal structures to have a tighter connection, even if they are not directly linked in the super-graph. By combining structured node content and topology information of networked graphs, our algorithm demonstrated good performance gain on real-world applications.

The key contribution of this article is threefold: (1) a new networked graph model for describing real-world objects containing structure information inside the object and between objects; (2) a mixed-similarity assessment considering structured content inside graphs and structure dependency between graphs; and (3) a new networked graph clustering method.

REFERENCES

Charu Aggarwal, Yuchen Zhao, and Philip Yu. 2010. On clustering graph streams. In *Proceedings of SIAM International Conference on Data Mining*.

Ben-Dor Amir, Ron Shamir, and Zohar Yakhini. 1999. Clustering gene expression patterns. *Journal of Computational Biology* 6, 3–4 (1999), 281–297.

R. Angelova and G. Weikum. 2006. Graph-based text classification: Learn from your neighbors. In *Proceedings of ACM SIGIR Conference*. 485–492.

Christopher M. Bishop (Ed.). 2006. *Pattern Recognition and Machine Learning*, vol. 1. Springer, New York, NY.

Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. 2003. Experiments on graph clustering algorithms. In *Proceedings of the 11th European Symposium on Algorithms*. 568–579.

T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. 1987. Graph bisection algorithms with good average case behavior. *Combinatorica* 7, 2 (1987), 171–191.

Deng Cai, Xiaofeng He, Jiawei Han, and Thomas S. Huang. 2011. Graph regularized non-negative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 8 (2011), 1548–1560.

Deng Cai, Xiaofei He, Xiaoyun Wu, and Jiawei Han. 2008. Non-negative matrix factorization on manifold. In *Proceedings of the 8th IEEE international Conference on Data Mining*. 63–72.

Yandong Cai, Nick Cercone, and Jiawei Han. 1990. An attribute-oriented approach for learning classification rules from relational databases. In *Proceedings of IEEE International Conference on Data Engineering (ICDE'90)*. 281–288.

J. J. M. Carrasco, D. C. Fain, and L. Zhukov K. J. Lang. 2003. Clustering of bipartite advertiser-keyword graph. In *Proceedings of the 3rd IEEE International Conference on Data Mining, Workshop on Clustering Large Data Sets*.

X. Chen and D. Cai. 2011. Large scale spectral clustering with landmark-based representation. In *Proceedings of AAAI Conference*. 313–318.

Yudong Chen, S. Sanghavi, and Xu Huan. 2014. Improved graph clustering. *IEEE Transactions on Information Theory* 60, 10 (2014), 6440–6455.

Hong Cheng, Yang Zhou, and Jeffrey Xu Yu. 2011. Clustering large attributed graphs: A balance between structural and attribute similarities. *ACM TKDD* 5, 2 (2011), Article No. 12.

Lianhua Chi, Bin Li, and Xingquan Zhu. 2013. Fast graph stream classification using discriminative clique hashing. In *Proceedings of Pacific Asian Conference on Knowledge Discovery and Data Mining (PAKDD'13)*.

F. Costa and K. De Grave. 2010. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning*.

Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing community structure identification. *Theory and Experiment* 9 (2005), P09008.

Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. 2005. A fast kernel-based multilevel algorithm for graph clustering. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 629–634.

Chris Ding and Tao Li. 2010. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 1 (2010), 45–55.

Luca Donetti and Miguel A. Muñoz. 2004. Detecting network communities: A new systematic and efficient algorithm. *Journal of Statistical Mechanics* 10 (2004), P10012.

Gary William Flake, Robert E. Tarjan, and Kostas Tsioutsiouliklis. 2004. Graph clustering and minimum cut trees. *Internet Mathematics* 1, 4 (2004), 385–408.

Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3 (2010), 75–174.

Santo Fortunato, Vito Latora, and Massimo Marchiori. 2004. A method to find community structures based on information centrality. *Physical Review E* 70, 5 (2004), 056104.

Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness rand efficient alternatives. In *Proceedings of Conference on Learning Theory (COLT*'03). 129–143.

Maria Giatsoglou and Athena Vakali. 2013. Capturing social data evolution using graph clustering. *IEEE Internet Computing* 17, 1 (2013), 74–79.

M. Girvan and M. E. J. Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99 (2002) 8271–8276.

Ting Guo and Xingquan Zhu. 2013. Understanding the roles of sub-graph features for graph classification: An empirical study perspective. In *Proceedings of ACM CIKM Conference*.

J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*. 281–297.

U. Kang, Hanghang Tong, and Jimeng Sun. 2012. Fast random walk graph kernel. In *Proceedings of the 12th SIAM international Conference on Data Mining*. 828–838.

Michihiro Kuramochi and George Karypis. 2001. Frequent subgraph discovery. In *Proceedings of the IEEE International Conference on Data Mining*. 313–320.

Si Quang Le, Tu Bo Ho, and T. E Hang Phan. 2004. A novel graph-based similarity measure for 2D chemical structures. *Genome Informatics* 15, 2 (2004), 82–91.

Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems* 13 (2001), 556–562.

X. F. Luo, Z. X., J. Yu, and X. Chen. 2011. Building association link network for semantic link on web resources. *IEEE Transactions on Automation Science and Engineering* 8, 3 (2011), 482–494.

Fragkiskos Malliaros and Michalis Vazirgiannis. 2013. Clustering and community detection in directed networks: A survey. *Physics Reports* 533, 4 (2013), 95–142.

M. E. J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69, 2 (2004), 026113.

A. Y. Ng, M. I. Jordan, and Y. Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems* 2 (2001), 849–856.

Younghee Park, Douglas S. Reeves, and Mark Stamp. 2013. Deriving common malware behavior through graph clustering. *Computers and Security* 39, B (2013), 419–430.

M. Rattigan, M. Majer, and D. Jensen. 2007. Graph clustering with network structure indices. In *Proceedings of International Conference on Machine Learning (ICML'07)*. 783–790.

Kaspar Riesen and Horst Bunke. 2010. *Graph Classification and Clustering Based on Vector Space Embedding*. World Scientific Publishing, Singapore.

Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo, and Koji Tsuda. 2009. gBoost: A mathematical programming approach to graph classification and regression. *Machine Learning* 75, 1 (2009), 69–89.

S. E. Schaeffer. 2007. Graph clustering. *Computer Science Review* 1 (2007), 27–64.

M. Seeland, S. A. Berger, A. Stamatakis, and S. Kramer. 2011. Parallel structural graph clustering. In *Proceedings of 2011 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*.

M. Seeland, A. Karwath, and S. Kramer. 2012. A structural cluster kernel for learning on graphs. In *Proceedings of ACM SIG KDD Conference*.

J. Shi and J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905.

A. Smalter, J. Huan, Y. Jia, and G. Lushington. 2010. GPD: A graph pattern diffusion kernel for accurate graph classification with applications in cheminformatics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 7, 2 (2010), 197–207.

A. Strehl and J. Ghosh. 2002. Cluster ensembles – A knowledge reuse framework for combining multiple partitions. *Machine Learning Research* 3 (2002), 583–617.

Y. Tian, R. A. Hankins, and J. M. Patel. 2008. Efficient aggregation for graph summarisation. In *Proceedings of ACM SIGMOD Conference*. 567–580.

Jacobo Torán. 2004. On the hardness of graph isomorphism. *SIAM Journal of Computing* 33, 5 (2004), 1093–1108.

Ulrik von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17 (2007), 395–416.

Hua Wang, Feiping Nie, Heng Huang, and Fillia Makedon. 2011. Fast nonnegative matrix tri-factorization for large-scale data co-clustering. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'11)*. 1553–1558.

X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. 2007. Scan: A structural clustering algorithm for networks. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD*'07). 824–833.

Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2014. GBAGC: A general Bayesian framework for attributed graph clustering. *ACM Transactions on Knowledge Discovery from Data* 9 (2014), Article No. 5.

Jaewon Yang, Julian McAuley, and Jure Leskovec. 2013. Community detection in networks with node attributes. In *Proceedings of the IEEE International Conference on Data Mining*.

Eng-Hui Yap, Tyler Rosche, Steve Almo, and Andras Fiser. 2013. Functional clustering of immunoglobulin superfamily proteins with protein-protein interaction information calibrated hidden Markov model sequence profiles. *Journal of Molecular Biology* 426, 4 (2013), 945–961.

X. Yin, J. Han, and P. S. Yu. 2005. Cross-relational clustering with user's guidance. In *Proceedings of ACM SIGKDD Conference*. 344–353.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. In *Proceedings of the VLDB Endowment*, vol. 2. 718–729.

Xiaoqu Zhu, Meilan Hu, Feng Zhang, Yu Tao, Chuming Wu, Shangzhu Lin, and Fule He. 2013. Expression profiling based on graph-clustering approach to determine colon cancer pathway. *Journal of Cancer Research and Therapeutics* 9, 3 (2013), 467–470.

Received February 2015; revised August 2016; accepted September 2016