Community Detection in Multi-Layer Graphs: A Survey

Jungeun Kim, Jae-Gil Lee*
Department of Knowledge Service Engineering, KAIST
Daejeon, Republic of Korea
{je_kim, jaegil}@kaist.ac.kr

ABSTRACT

Community detection, also known as graph clustering, has been extensively studied in the literature. The goal of community detection is to partition vertices in a complex graph into densely-connected components so-called *communities*. In recent applications, however, an entity is associated with multiple aspects of relationships, which brings new challenges in community detection. The multiple aspects of interactions can be modeled as a *multi-layer graph* comprised of multiple interdependent graphs, where each graph represents an aspect of the interactions. Great efforts have therefore been made to tackle the problem of community detection in multi-layer graphs. In this survey, we provide readers with a comprehensive understanding of community detection in multi-layer graphs and compare the state-of-the-art algorithms with respect to their underlying properties.

1. INTRODUCTION

Graph mining in complex networks has attracted significant attention during the past several years. One of the important tasks in graph mining is community detection, in which the objective is to partition a graph into several densely-connected components. Such components correspond to sets of similar vertices, and can thus be regarded as a *community* [17]. Since this problem arises in a broad range of applications, a large number of approaches have been proposed in the literature [10, 13, 15].

In contrast to the traditional problem, recent applications, such as mobile and social network analyses, give rise to intriguing new challenges [6]. In this context, assumably, data encapsulates multiple aspects of human interactions, *e.g.*, those among coworkers and those among friends. The multiple aspects of relationships can be represented by a multi-layer graph comprised of multiple interdependent graphs, where each graph represents an aspect of the relationships. Therefore, great efforts have been made to solve the challenge of community detection in multi-layer graphs.

The goal of this survey is to provide a timely remark on the status of improving community detection in multi-layer graphs. We offer a brief overview of primary algorithms and classify them with respect to their underlying strategies.

The rest of this paper is organized as follows. Section 2 discusses the background information regarding multi-layer graphs. Section 3 presents multi-layer graph datasets used in recent studies. Section 4 introduces community detection approaches in two-layer graphs. Section 5 introduces community detection approaches in multi-layer graphs. Section 6 presents comparisons of community detection approaches in multi-layer graphs. Section 7 suggests promising future research directions in multi-layer graphs. Finally, Section 8 concludes this survey.

2. BACKGROUND

In this section, we discuss some background information about multi-layer graphs. We present the formal definitions of multi-layer graphs [14] in Section 2.1. Then, we briefly summarize the community detection approaches for single graphs and the challenges for developing those for multi-layer graphs in Section 2.2. To enhance the readability of this survey, frequently used symbols are summarized in Table 1.

Table 1: The summary of symbols.

Symbol	Description
G	a graph
V	a set of vertices
S	a set of attributes
L	a set of layers
n	the number of vertices
m	the number of edges
k	the number of clusters
t	the number of attributes
l	the number of layers

*Jae-Gil Lee is the corresponding author.

2.1 Multiple Network Models

2.1.1 Multi-Layer Graphs

The definition of a multi-layer graph depends on that of a single-layer graph.

Definition 1. [14] A *single-layer graph* is a weighted graph (V, w) where V is a set of vertices and w is a set of edge weights: $(V \times V) \rightarrow [0,1]$.

Figure 1 shows an example of a single undirected graph (without specifying the edge weights). Assume that it is a subgraph of Facebook's network. Each vertex represents the user, and each edge denotes the relationship between users. The weight of the edge is the strength of the relationship.

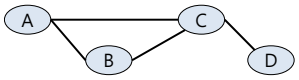


Figure 1: A single-layer graph.

When we start characterizing multi-layer graphs, understanding which vertices in one graph correspond to vertices in the other is important because the multi-layer graph is comprised of multiple *interdependent* graphs. A *node mapping* can formalize this task.

Definition 2. [14] A *node mapping* from a graph layer $L_1 = (V_1, w_1)$ to another graph layer $L_2 = (V_2, w_2)$ is a function $f : V_1 \times V_2 \rightarrow [0, 1]$. For each $u \in V_1$, the set $C(u) = \{v \in V_2 | f(u, v) > 0\}$ is the set of V_2 vertices corresponding to u.

Figure 2 illustrates an example of a multi-layer graph. Assume that layer 1 is the Facebook network and layer 2 is the Twitter network. If the users in the Facebook network also have an account on Twitter, then the Twitter network can be used to represent these users and their relationships. Note that every user can be identified by one account on each layer. This graph is generally called a *pillar multi-layer graph* since every user can be seen as a pillar traversing every layer denoting the level of physical reality [14]. A pillar multi-layer graph is formally defined by node mapping, $|C(u)| \in \{0, 1\}$.

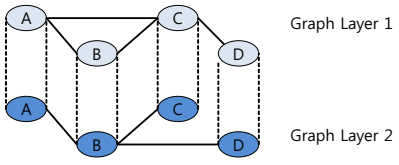


Figure 2: A pillar multi-layer graph.

A generic multi-layer graph is formally defined based both on a set of single layers and a matrix of node mappings.

Definition 3. [14] A *multi-layer graph* is a tuple $MLN = (L_1, ..., L_l, IM)$ where $L_i = (V_i, w_i), i \in 1, ..., l$ are graph layers and IM (Identity Mapping) is an $l \times l$ matrix of node mappings, with $IM_{i,j} : V_i \times V_j \rightarrow [0, 1]$.

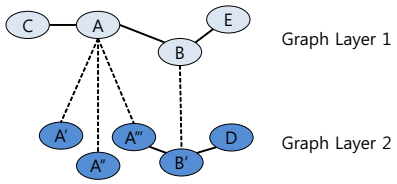


Figure 3: A general multi-layer graph.

Figure 3 shows an example of a multi-layer graph that is more complex than a pillar multi-layer graph. For example, in Figure 3, on layer 1, A is an account of a user in FriendFeed, and A′, A″, and A‴ on layer 2 are the social media accounts that the user has registered. We call this network a *general multi-layer graph*. Note that a vertex in one graph layer corresponds to multiple vertices in another. This case is typically shown in *social media aggregators*, such as FriendFeed, which support various social network services with a single access point as long as a user has registered for those services. Thus, vertices do not necessarily denote users, but more generally, accounts.

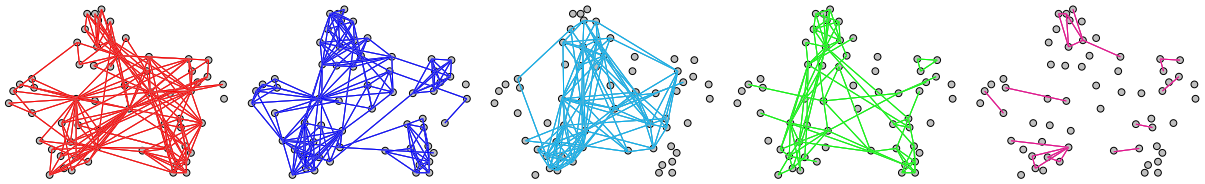
2.1.2 Heterogeneous Information Networks

The definition of a heterogeneous information network depends on that of an information network.

Definition 4. [23, 24] An *information network* is defined as a directed graph $G = (V, E)$ with an object type mapping function $\phi : V \rightarrow \mathcal{A}$ and a link type mapping function $\psi : E \rightarrow \mathcal{R}$, where each object $v \in V$ belongs to one particular object type $\phi(v) \in \mathcal{A}$, and each link $e \in E$ belongs to a particular relation $\psi(e) \in \mathcal{R}$.

If the number of object types $|\mathcal{A}| > 1$ or the number of link types $|\mathcal{R}| > 1$, the network is called a *heterogeneous information network* [23, 24]. A bibliographic information network is a typical example, containing objects from four types of entities: papers, venues, authors, and terms. Each paper has distinct types of links to a set of authors, a venue, a set of words, a set of citing papers, and a set of cited papers, respectively.

A heterogeneous information network can be translated to a *general* multi-layer graph in Definition 3, and vice versa. More specifically, an object type corresponds to a layer L_i, the links within an object type correspond to w_i, and the links between



(a) Work. (b) Lunch. (c) Facebook. (d) Friend. (e) Coauthor.

Figure 4: An example of a multi-layer graph called the AUCS dataset.

different object types correspond to $IM_{i,j}$. That is, these two definitions are syntactically equivalent.

Despite this equivalence, the two definitions are actually being used for slightly different meanings. General multi-layer graphs emphasize *multiple types of relationships* between similar types of entities. For example, in Figure 3, all the entities are social media accounts. On the other hand, heterogeneous information networks emphasize *heterogeneous types of entities* connected by different relationships. Overall, we rely on the typical meaning of multi-layer graphs in this paper.

2.2 Current Status and Challenges

2.2.1 Community Detection in Single Graphs

Many community detection approaches have been proposed for single-layer graphs. Fortunato [7] and Schaeffer [19] conducted really extensive survey on this topic. Representative algorithms include graph partitioning algorithms, modularity-based algorithms, spectral algorithms, and structure definition algorithms [7, 19]. The objective of *graph partitioning algorithms* is to divide the vertices such that cut size is minimal. *Cut size* is determined by the number of edges lying between partitions. The goal of *modularity-based algorithms* is to partition the vertices such that modularity is maximal. *Modularity* is defined by the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random. *Spectral algorithms* partition the graph into communities using the eigenvectors of graph matrices. A graph Laplacian matrix is typically used for the graph matrix. *Structure definition algorithms* discover communities such that a very strict structural property is satisfied. In other words, they find communities satisfying the meta definitions of a community such as k-clique, r-quasi clique, and s-plex.

2.2.2 Challenges for Multi-Layer Graphs

In contrast to the community detection problem in single graphs, new challenges arise for community detection in multi-layer graphs. Intuitively, each single layer has a piece of meaningful information from its own perspective; however, one can expect improved community detection results through the proper and efficient merging of information in each layer. Thus, an important open question is how to exploit and fuse the multiple aspects of information to generate improved understanding of vertices and their relationships. In addition, since we are confronted with managing multiple layers (often called *networks of networks*), scalability remains a significant challenge because of the larger resulting search spaces [2].

3. MULTI-LAYER GRAPH DATASETS

In this section, we introduce various multi-layer graph datasets. Figure 4 illustrates an example of a multi-layer graph called the AUCS dataset. In this graph, the multiple layers represent relationships between 61 employees of a University department in five different aspects: (i) coworking, (ii) having lunch together, (iii) Facebook friendship, (iv) offline friendship (having fun together), and (v) coauthorship. Popular datasets used in academic papers are as follows. Note that some layers are constructed *by using attribute information*. In these cases, an edge between two vertices is formed if attribute similarity is higher than a given threshold. This list is also available at `http://dm.kaist.ac.kr/datasets/multi-layer-network/`.

- **MIT Reality Mining [6]**
  This is a mobile phone dataset including 87 users on the MIT campus. Each layer represents the relationships defined by physical locations, bluetooth scans, and phone calls, respectively.
- **Enron Email [17]**
  This is an email message dataset between employees of the Enron corporation. It contains 200,399 messages belonging to 158 members. One layer contains the relationships defined by the existence of email communications, and the other contains those defined by the similarity of text messages.
- **Mobile Phone [6]**
  This is a mobile phone dataset collected by Nokia Research Center(NRC) Lausnne [8]. It contains

about 200 mobile users in Lausanne, Switzerland. Each layer represents the relationships defined by physical locations, bluetooth scans, and phone calls, respectively.

- **Cora [6]**
  This is a bibliographic dataset including 292 research papers. Layers represent three different research fields such as natural language processing, data mining, and robotics, respectively.
- **IMDB [1]**
  This is a movie database managed by IMDB, which contains 300 vertices and 18,368 edges. Vertices represent actors and edges are formed if two actors worked together. In this dataset, there exists four layers: (i) the first year of collaboration, (ii) the last year of collaboration, (iii) the average incomes, and (iv) the average number of sold tickets. In other words, four layers have the same edges but different edge labels.
- **Airline Transportation Multiplex [3]**
  This is a network composed of the airline operating in Europe. It contains 450 vertices and 3,588 edges. This data includes total thirty-seven layers and each one corresponds to a different airline.
- **SIAM Journal [25]**
  This is a bibliographic dataset containing 5,022 vertices which are papers. Five layers are formed from five different similarity matrices. First three are defined by the text similarity based on the abstract, title, and keyword, respectively. The other two are obtained by the number of common authors between papers and the citation relation.
- **Political Blogs [26] [28]**
  This is a webblog network on US politics, which contains 1,490 vertices and 19,090 edges. Vertices represent webblogs, and edges hyperlinks between webblogs. Each blog in the dataset has an attribute denoting its political position as either liberal or conservative. Thus, one layer depicts explicit hyperlinks, and the other depicts political preferences.
- **CiteSeer [12] [18] [21]**
  This is a citation network of computer science publications containing 3,312 vertices and 4,536 edges. One layer contains the relationships defined by citation, and the other contains those defined by content similarity.
- **US Stock Market [27]**
  This is a US stock market graph database containing 11 graph layers. On average it contains 3,636 vertices and 206,747 edges. Each layer is a graph made by setting the different correlation coefficient value based on stock price.
- **Arxiv Publication Database [28]**
  This is a bibliographic dataset including 13,396 vertices and 673,800 edges. Each layer corresponds to citation relationships with different research topics. Thus, the number of layers is equivalent to that of topics.
- **Flickr [17] [18]**
  This is a social network with tagged photos including 16,710 vertices and 716,063 edges. Each vertex represents a user, and the edge exists if the user is in another's contact list or if they favor the same images. In other words, one layer represents the relationships defined by the explicit contact list, and the other represents those defined by the common interest retrieved from photo sharing between two users.
- **DBLP [1] [20] [26] [28]**
  This is a bibliographic dataset including up to 108,030 vertices and 276,658 edges. A vertex stands for an author, and an edge is formed if two authors write a research paper together or share the same research interest. Using this dataset, we can make a two-layer graph as well as a general multi-layer graph whose layers are more than 2. In the two-layer graph, one layer contains the relationships defined by coauthorship while the other contains those defined by the sameness of the research interest. In the general multi-layer graph, each layer corresponds to the coauthorships in the different venues (conferences or journals).
- **LastFm [20]**
  This is a social music network which consists of 272,412 vertices and 350,239 edges. One layer represents the relationships based on friendships between users, and the other represents those based on the sameness of the musical tastes.
- **Higgs Twitter [5]**
  This is the multiplex of social interactions in Twitter including 456,631 vertices and 16,070,185 edges. Each layer represents friendship, replying, mentioning, and retweeting, respectively.
- **Wikipedia [18]**
  This dataset is from the static dump of English Wikipedia pages. It consists of 3,580,013 vertices and 162,085,383 edges. One layer contains the relationships defined by explicit page links, and the other contains those defined by text similarity between pages.

Table 2: The summary of multi-layer datasets.

No.	Name	# Vertices	# Edges	# Layers	Type	Publicly Available
1	AUCS	61	620	5	pillar	Y [1]
2	MIT Reality Mining [6]	87	-	3	pillar	Y [2]
3	Enron Email [17]	158	200,399	2	pillar	Y [3]
4	Mobile Phone [6]	200	-	3	pillar	N
5	Cora [6]	292	-	3	pillar	Y [4]
6	IMDB [1]	300	18,368	4	pillar	△ [5]
7	Airline Transportation Multiplex [3]	450	3,588	37	pillar	Y [6]
8	SIAM Journal [25]	5,022	-	5	pillar	N
9	Political Blogs [28] [26]	1,490	19,090	2	pillar	Y [7]
10	CiteSeer [12] [18] [21]	3,636	4,536	2	pillar	Y [8]
11	US stock market [27]	3,312	206,747	11	pillar	N
12	Arxiv publication [28]	13,396	673,800	7	pillar	N
13	Flickr [17] [18]	16,710	716,063	2	pillar	Y [9]
14	DBLP [1] [20] [26] [28]	108,030	276,658	various	pillar	Y [10]
15	LastFm [20]	272,412	350,239	2	pillar	△ [11]
16	Higgs Twitter [5]	456,631	16,070,185	4	pillar	Y [12]
17	Wikipidia [18]	3,580,013	162,085,383	2	pillar	N
18	FriendFeed [4]	9,717,499	15,000,000	various	general	Y [13]

- **FriendFeed [4]**
  This is one of social media aggregators. It contains about 400,000 users and 1 million posts with 15 million subscription relationships. In general, vertices stand for users, and edges various relationships between users. On the other hands, vertices can also be posts, and edges relationships between users and posts. Thus, layers can be various, for example, the types of services as well as the different relationships between users and posts.

Table 2 shows a brief summary of the multi-layer network datasets. If certain information of datasets does not exist in the reference papers, we fill in the blank with "−". For the "public available" column, if we can directly get the dataset through the web, we assign "Y". If we need extra efforts (*e.g.*, crawling) to get datasets, we assign "△".

4. COMMUNITY DETECTION IN TWO-LAYER GRAPHS

In this section, we introduce community detection algorithms in two-layer graphs. All algorithms described in this section can only support two-layer graphs and mostly consider structural and attribute information. One layer represents the original topology of a graph as structural information, and the other layer is derived by calculating the similarity between the vertices based on their attribute information. Such graphs with additional attribute information do not seem to conform to the definition multi-layer graphs. However, they have been regarded as a typical case of two-layer graphs since attribute information can be easily transformed to a layer—*e.g.*, by creating an edge between vertices if the attribute similarity between them is above a certain threshold. Thus, we categorize such graphs into two-layer graphs.

4.1 Cluster Expansion

Li *et al.* [12] proposed a hierarchical community detection algorithm based on both relations and textual attributes using the cluster expansion philosophy. This algorithm focuses on quickly finding initial cores as seeds of communities and expanding the cores into the communities in order to enhance scalability. In this paper, the CiteSeer dataset (No.10 in Table 2) was used. In the No.10 dataset, one layer represents the citation relationship between papers, and the other represents the degree of content similarity of the titles and abstracts of papers.

The algorithm consists of four major steps: core probing, core merging, affiliation, and classification. Figure 5 shows an overview of the algorithm (without specifying attribute information).

First, structural information is used solely to find cores, denoted as K_i, using the frequent itemset mining method derived from the Apriori algorithm.

[1] http://sigsna.net/impact/datasets/
[2] http://realitycommons.media.mit.edu/index.html
[3] http://bailando.sims.berkeley.edu/enron_email.html
[4] http://www.cs.umass.edu/~mccallum/data.html
[5] http://imdb.com
[6] http://complex.unizar.es/~atnmultiplex/
[7] http://networkdata.ics.uci.edu/data.php?id=102
[8] http://www.cs.umd.edu/projects/linqs/projects/lbc/index.html
[9] http://staff.science.uva.nl/~xirong/index.php?n=DataSet.Flickr3m
[10] http://informatik.uni-trier.de/~ley/db/
[11] http://www.last.fm
[12] http://www.plexmath.eu/?page_id=320/
[13] http://sigsna.net/impact/datasets/

After the set of all outgoing relations is listed for each document, the process of finding cores can be transformed into that of computing frequent itemsets. Each core will be used as a community seed. This step will enhance the scalability of the subsequent steps since the analysis scope is limited to each core. Then, cores are merged based on textual analysis using text similarity (*i.e.*, attribute information). In the core merging step of Figure 5, K_3 and K_4 are merged since they are linked and also topically relevant (not shown in the figure). In the affiliation step, initial communities are constructed through relation propagation. For each vertex v_i in a cluster C, the algorithm finds all vertices that are adjacent to v_i and adds them to C. Now every merged K_i is expanded to C_i in Figure 5. Since finding communities based solely on relation propagation may generate false hits, communities are refined based on classification using attribute analysis. In this step, LDA is used to reduce dimensionality, and all vertices are transformed into the feature vectors to represent their topical positions. Then, vertices are classified based on the SVM, and negatively labeled vertices are removed. For example, v_D is dropped from C_1.

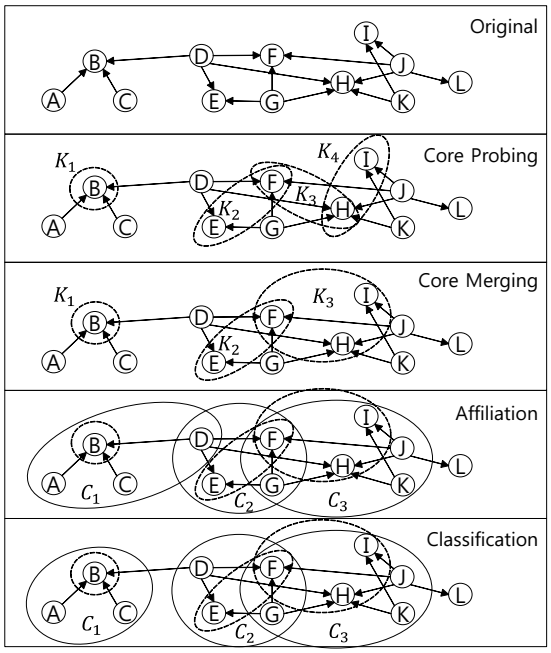


Figure 5: The overview of the community discovery algorithm [12].

4.2 Matrix Factorization

Qi *et al.* [17] proposed a community detection algorithm based both on link structure and edge content using the Edge-Induced Matrix Factorization (EIMF). In this paper, the Enron email and Flickr datasets (No.3 and No.13 in Table 2) were used. In the No.13 dataset, one layer depicts the relationships defined by the contact list, and the other depicts those defined by the favorite photo shared by the users.

The main contribution of this algorithm is using edge content for the community detection process. Edge content can be a useful source of information when nodes interact with multiple communities, since it can assist in distinguishing between the different interactions of nodes. Figure 6 shows an example of an edge-based social network in the No.13 dataset. Intuitively, edges can be divided into two different groups, such as a family (AB, BC, CD, AD) and people with similar musical interests (AE, AF). Moreover, it is clear that the user A belongs to both communities based on the edge content, whereas the same finding is unclear in terms of a vertex-centric perspective.

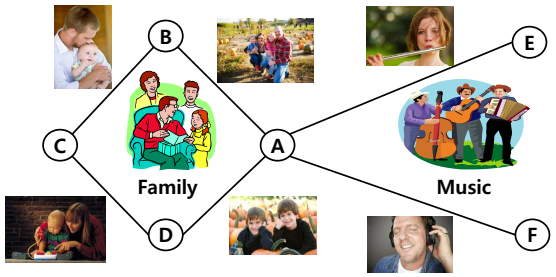


Figure 6: An example of an edge-based social media network [17].

This algorithm consists mainly of two parts: the EIMF based purely on the link structure, and incorporation of the edge content into the EIMF.

In the first part, an incidence matrix is formed using the link structure. Then, the latent edge matrix E is constructed from the incidence matrix using matrix factorization, which is obtained by minimizing Eq.(1).

$$O^l(E) = \| E^T \cdot E \cdot \triangle - \Gamma \|_F^2 \qquad (1)$$

Here, E is a $k \times m$ matrix with each column corresponding to a k-dimensional feature vector for an edge, Γ is an $m \times n$ incident matrix, and Δ is a normalization of the incident matrix such that every column-wise sum becomes 1. Then, by the definition of matrix factorization, Eq.(1) indicates the error of the approximation by E when compared with the link structure Γ. Since each column of E represents the membership of the edge to k communities, this procedure of matrix factorization can be regarded as a community detection technique using the link structure.

In the second part, two approaches are proposed in order to consider the edge content by way of reflecting the similarity among the edge content in matrix factorization. The former approach is to optimize Eq.(2).

$$O(E) = O^l(E) + \lambda \cdot O^c(E) \tag{2}$$

Here, $O^c(E)$ denotes the error of the approximation by E when compared with the similarity of edge content instead of the link structure, and λ is a weighting factor to consider the degree of importance of the link structure and edge content. The other approach is developed to avoid the necessity of tuning the parameter λ, and the reader can refer to [17] for the details.

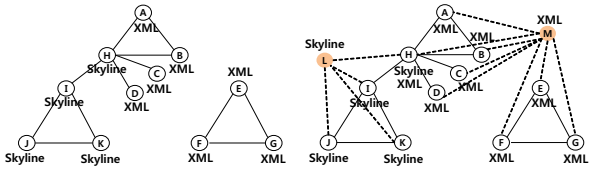
4.3 Unified Distance

Zhou *et al.* [28] proposed a community detection algorithm, called SA-Cluster, based on both structural and attribute similarities using a unified distance measure. In this paper, political blogs and the DBLP datasets (No.9 and No.14 in Table 2) were used. In the No.14 dataset, one layer represents the relationships created by coauthorship between researchers, and the other layer represents those defined by the similarity of research interests.

The main contribution of SA-Cluster is twofold: (1) a unified distance measure to fuse structural and attribute similarities; (2) a weight self-adjustment method to modulate the degree of importance of structural and attribute similarities.

First, the unified distance measure is formulated based on the *attribute-augmented graph* using the Random Walk with Restart (RWR). Figure 7a shows the original coauthor network, and Figure 7b shows the attribute-augmented graph with research topics. In the attribute-augmented graph, *attribute vertices* are added to represent attribute values, and the original vertices are connected to the corresponding attribute vertices. For example, the research topics, "Skyline" and "XML", are added as attribute vertices (two shaded vertices L and M in Figure 7b). Then, the researchers are connected via attribute vertices if they are interested in the same research topic. Intuitively, the larger the number of common attribute values between two vertices, the higher the degree of similarity between the two vertices, since more random walk paths can exist.

Second, the graph clustering algorithm that follows k-medoids clustering is performed based on the unified distance measure. More importantly, weight self-adjustment is conducted in each iteration of the algorithm. The weight of an attribute a_i in the



(a) A coauthor graph. (b) An augmented graph.

Figure 7: A coauthor network example with a topic attribute [28].

$(t+1)^{th}$ iteration is computed as Eq.(3).

$$w_i^{t+1} = \frac{1}{2}(w_i^t + \Delta w_i^t) \tag{3}$$

The weight increment Δw_i is measured by a majority voting mechanism. The voting mechanism counts the number of vertices within clusters that share the same attribute values for estimating clustering tendency of the attribute, and then adjusts the attribute weight. That is, if a large number of vertices within clusters have the same value of an attribute a_i, it denotes that a_i has high clustering tendency and increases the weight w_i of a_i accordingly.

4.4 Model-Based Method

Xu *et al.* [26] proposed a model-based community detection approach based on both structural and attribute aspects of a graph. In this paper, the datasets and graph layers were the same as those used in the authors' previous work [28].

The key point of this approach is the use of a *probabilistic model* that fuses both structural and attribute information instead of an artificial distance measure. The algorithm consists of two major parts: the construction of the probabilistic model and a variational approach to solve the model.

In the first part, a Bayesian probabilistic model is proposed for community detection over a clustered attributed graph. The clustered attributed graph used in this model is represented by $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$, where $\mathbf{X} = [X_{ij}]$ is an $n \times n$ adjacency matrix, $\mathbf{Y} = [Y_t^i]$ is an $n \times t$ attribute matrix, and $\mathbf{Z} = [Z_i]$ is a $n \times 1$ cluster vector that contains the label of a given vertex's cluster. This model defines a joint probability distribution $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ for all possible communities and attributed graphs. α, θ, ϕ are parameters for generating the probabilistic model, where α denotes the vertex distribution of each cluster, θ implies the attribute distribution of each cluster, and ϕ denotes the edge occurrence probabilities between clusters.

Based on the model, the problem of community detection is transformed into a probabilistic inference problem, finding the maximum-a-posteriori

(MAP) configuration of communities $\mathbf{Z}$ with conditions $\mathbf{X}$ and $\mathbf{Y}$, as formulated by Eq.(4).

$$\mathbf{Z}^* = \underset{\mathbf{Z}}{\operatorname{argmax}}\, p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) \tag{4}$$

However, it is computationally infeasible to find the global maximum for a large set of $\mathbf{Z}$.

In the second part, a variational algorithm is introduced to solve the probabilistic inference problem. The major principle is to approximate the distribution $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ using a variational distribution $q(\alpha, \theta, \phi, \mathbf{Z})$. Additionally, if we restrict the variational distribution to a family of distributions that factorize as Eq.(5), finding the global maximum translates as finding the local maximum in Eq.(6). Please refer to [26] for the details of the mathematical derivations.

$$q(\alpha, \theta, \phi, \mathbf{Z}) = q(\alpha)q(\theta)q(\phi)\prod_i q(Z_i) \tag{5}$$

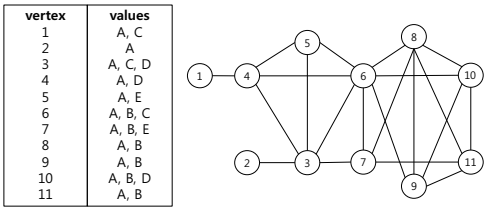
$$\begin{aligned}\mathbf{Z}^* &= \underset{\mathbf{Z}}{\operatorname{argmax}}\, p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) \\ &= [\underset{Z_1}{\operatorname{argmax}}\, q(Z_1), \underset{Z_2}{\operatorname{argmax}}\, q(Z_2), ..., \underset{Z_N}{\operatorname{argmax}}\, q(Z_N)]\end{aligned} \tag{6}$$

4.5 Pattern Mining

Silva *et al.* [21] proposed a community detection algorithm based on structural correlation pattern mining, called SCPM. In this paper, the CiteSeer, DBLP, and LastFm datasets (No.10, No.14, and No.15 in Table 2) were used. In the No.15 dataset, one layer contains friendships between users, and the other contains their shared musical preferences, *e.g.*, favorite singers.

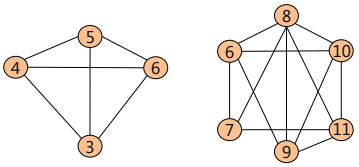
The main contribution of SCPM is to uncover the interaction between vertex attributes and dense subgraphs using both frequent itemset mining and quasi-clique mining. Here, a dense subgraph is defined by a γ-quasi-clique. The *structural correlation pattern* is formed if the proportion of the vertices in the dense subgraph that contain a given set of attribute values is above a threshold. In more detail, the algorithm first finds a frequent itemset S (*i.e.*, a set of attribute values appearing together in many vertices) from the entire graph G and obtains the subgraph G' induced by S. Then, it identifies a γ-quasi-clique Q from G'. Finally, the *structural correlation* of S is calculated by checking whether each vertex in G' belongs a quasi-clique Q. A structural correlation pattern should preserve a high value of structural correlation.

Figure 8 shows a toy example of SCPM. Figure 8a contains a set of attribute values for each vertex as well as an entire graph; Figure 8b depicts two examples of dense graphs. For a frequent itemset {A,B}, a subgraph {6,7,8,9,10,11} is induced from the entire graph, since these vertices include {A,B}. Then, a dense graph (the second one in Figure 8b) is obtained from this subgraph. Last, ({A,B}, {6,7,8,9,10,11}) is a structural correlation pattern with structural correlation 1, implying that the value set {A,B} appears on every vertex of the subgraph {6,7,8,9,10,11}.



vertex	values
1	A, C
2	A
3	A, C, D
4	A, D
5	A, E
6	A, B, C
7	A, B, E
8	A, B
9	A, B
10	A, B, D
11	A, B

(a) The graph with vertex attributes.



(b) The dense subgraphs.

Figure 8: An example of structural correlation pattern mining [21].

However, simply combining frequent itemset mining and quasi-clique mining will suffer from high computational overhead since the two problems are known to be #P-hard. Thus, two pruning techniques are proposed: (1) vertex pruning and (2) candidate set pruning. The former eliminates vertices that do not belong to quasi-cliques in the graph derived by a given attribute-value set or any quasi-clique in each iteration. The latter excludes candidate sets after the $(i+1)^{th}$ step if they do not satisfy the condition in the i^{th} step.

4.6 Graph Merging

Ruan *et al.* [18] proposed a community detection approach, called CODICIL, to combine structural and attribute information using the graph merging process. In this paper, the Wikipedia, Flickr, and CiteSeer datasets (No.17, No.13, and No.10 in Table 2) were used. In the No.17 dataset, one layer represents explicit hyperlinks, and the other represents content similarities.

The main contribution of this algorithm is to strengthen the community signal by eliminating noise in the link structure using content information. Figure 9 shows the work flow of the proposed approach. This approach consists of four steps: creating content edges, combining edges, sampling edges with bias, and clustering. First, for each vertex v_i, its k most content-similar neighbors are computed by calculating cosine similarity. Then, con-

tent edges are formed between the vertex v_i and its top-k neighbors. Second, the newly-created content edge set and the original topological edge set are simply unified. Third, for each vertex v_i, the edges to retain are selected from its local neighborhood based on either cosine similarity or Jaccard similarity. Last, clustering is performed on the merged graph. Since the process of merging graphs is performed independently of community detection algorithms, any conventional community detection algorithms can be applied.

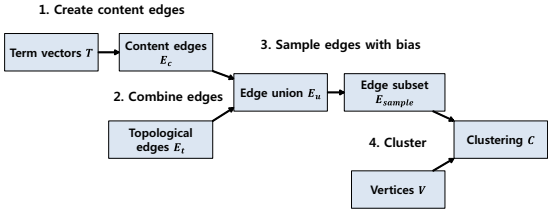


Figure 9: The work flow of CODICIL [18].

5. COMMUNITY DETECTION IN MULTI-LAYER GRAPHS

In this section, we introduce community detection algorithms that can support multi-layer graphs containing *more than or equal to two* layers.

5.1 Matrix Factorization

Tang *et al.* [25] and Dong *et al.* [6] proposed graph clustering algorithms for multi-layer graphs based on matrix factorization. In these papers, the MIT Reality Mining, mobile phone, Cora, and SIAM Journal datasets (No.2, No.4, No.5, and No.8 in Table 2) were used. In the No.2 and No.4 datasets, the layers represent the relationships defined by physical locations, bluetooth scans, and phone calls, respectively. In the No.5 dataset, the layers contain three different research domains: natural language processing, data mining, and robotics. In the No.8 dataset, the layers depict five different similarity matrices retrieved from the abstract, title, keywords, author, and citation fields.

The main idea of these two algorithms is to fuse different information by extracting common factors from multiple layers, which may then be used by general clustering methods. The major difference is that Tang *et al.* [25] approximates *adjacency matrices* while Dong *et al.* [6] approximates *graph Laplacian matrices*. To achieve this goal, they approximate each layer through a low-rank matrix factorization $O \approx P\Lambda P^t$, where O is an object matrix they try to approximate, which is either an adjacency matrix or a Laplacian matrix, P is an $n \times n$ eigenvector matrix, and Λ is an $n \times n$ eigenvalue matrix. When multiple layers are being considered, O is naturally extended to $O^{(i)}$, for $i = 1, \ldots, l$. Also, a common factor matrix should be reflected by the multiple factorizations. Hence, the objective function is defined as minimizing Eq.(7), where P is an $n \times n$ matrix representing the common factor of all layers, $\Lambda^{(i)}$ is an $n \times n$ matrix capturing the characteristics of i^{th} layer, $||\cdot||$ is the Frobenius norm, and α is a regularization parameter.

$$G = \frac{1}{2}\sum_{i=1}^{l} ||O^{(i)} - P\Lambda^{(i)}P^T||_F^2 + \frac{\alpha}{2}(\sum_{i=1}^{l} ||\Lambda^{(i)}||_F^2 + ||P||_F^2) \quad (7)$$

However, the solution of this objective function is not jointly convex in P and $\Lambda^{(i)}$. Thus, they proposed an alternative method that transforms the problem of finding the global minimum into that of finding the local minimum. In brief, they first fix P and optimize $\Lambda^{(i)}$, and then fix $\Lambda^{(i)}$ and optimize P. This procedure is repeated until the solution converges.

5.2 Pattern Mining

Zeng *et al.* [27] proposed a subgraph mining algorithm for finding quasi-cliques that appear on multiple layers with a frequency above a given threshold. In this paper, the US stock market database (No.11 in Table 2) was used. In the No.11 dataset, each layer represents a graph formed by different correlation coefficient values in terms of stock prices.

The main contribution of this algorithm is to find *cross-graph quasi-cliques* in a multi-layer graph that are frequent, coherent, and closed. Generally, the cross-graph quasi-clique has been defined as a set of vertices belonging to a quasi-clique that appears on all layers and must be the maximal set [16]. However, this algorithm does not limit the minimum support to be 100%, meaning that it attempts to find quasi-cliques on above a certain percentage of the layers in a multi-layer graph. The final output does not contain a quasi-clique Q if any superset of Q forms a quasi-clique with the same support, because the output must be closed.

To satisfy this goal, the algorithm first converts the subgraphs into their canonical forms. Since the algorithm does not take the exact topology of a quasi-clique into account as long as it satisfies given properties, the subgraph can be represented by the minimum string with the assumption that all vertices have the total order. Then, the algorithm enumerates feasible candidates for γ-quasi-cliques by using the DFS strategy with pruning techniques. Finally, the algorithm selects *closed* γ-quasi-cliques based on the closure-checking scheme. A naive approach of the closure-checking scheme

scans all γ-quasi-cliques, and then checks whether those quasi-cliques can be subsumed by other quasi-cliques. Since this naive approach is very costly, the algorithm adopts an efficient variational approach using the enumeration tree satisfying the condition that a descendant must subsume an ancestor. The key principle of the variational approach is to conduct the closure checking for each quasi-clique Q after all of its descendants have been processed.

Boden *et al.* [1] proposed a graph clustering algorithm in multi-layer graphs with edge labels, called MiMAG. In this paper, the IMDM, Arxiv, and DBLP datasets (No.6, No.12, and No.14 in Table 2) were used. In the No.6 dataset, each layer depicts different information about movies in which two actors star together. In the No.12 or No.14 datasets, each layer represents the citation or coauthorship relationships in different topics or conferences.

The main contribution of MiMAG is to find clusters, called MLCS (Multi-Layer Coherent Subgraph), satisfying both aspects of structural density and edge label similarity. In order to achieve the structural density of MLCS, a γ-quasi-clique model is used. For the edge label similarity of MLCS, a cell-based cluster model is used. Putting them together, the algorithm finds the densely-connected subgraphs whose edge labels vary at most by a certain threshold w. Such a subgraph is called an MLCS when it satisfies the two conditions on at least two layers.

However, listing all MLCSs produces numerous similar clusters, possibly containing redundant information, since MiMAG allows MLCSs to overlap with each other. For example, in Figure 10, the clusters C_2 and C_3 are redundant since they share a large number of the same vertices, *i.e.*, $\{f, g, h\}$, on layer 1.

In order to avoid redundancy, a redundancy relation is introduced [1]. It defines a cluster C to be redundant with respect to a cluster C' if the edges of C and those of C' overlap at a high rate and the quality of C' is higher than that of C. The quality

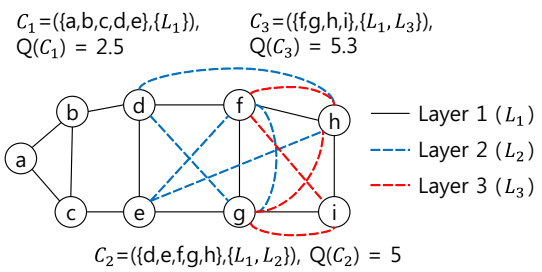


Figure 10: An example of overlapping clusters [1].

of a cluster $C = (V, L)$ is defined as Eq.(8), where V is a set of vertices, L denotes a set of layers, and $\gamma_L(V)$ represents the average density of the cluster on L.

$$Q(C) = \begin{cases} |V| \cdot |L| \cdot \gamma_L(V), \text{ if } |V| \geq 8 \wedge |L| \geq 2 \\ -1, \text{ otherwise} \end{cases} \tag{8}$$

Thus, MiMAG prefers the clusters that contain more vertices, contain more edges (*i.e.*, denser), and appear on more layers. In Figure 10, it is formally defined that C_2 is redundant with respect to C_3.

6. COMPARISON

In this section, we compare ten community detection algorithms introduced in Sections 4 and 5 with respect to the following seven properties. When selecting properties, we refer to the popular properties of subspace clustering [9] since community detection in multi-layer graphs resembles subspace clustering in considering that both methods deal with multiple dimensions of datasets. Among the properties in [9], only those closely related to community detection in multi-layer graphs are selected. Then, if a property is satisfied by none of the algorithms in this paper, we exclude it. Overall, P.1∼P.7 except P.4 correspond to a subset of the properties in [9]. P.4 is inspired by a widely-known categorization of attribute selection: the filter model and the wrapper model [22].

- **Property 1: Multiple layer ($l \geq 2$) applicability**
  All algorithms we introduced are designed for the community detection problem in multi-layer graphs. However, some algorithms support only a two-layer graph, while the others support a multi-layer graph containing more than two layers.
- **Property 2: Consideration of each layer's importance**
  Since each aspect of relationships may have different importance in the real world, considering the importance of each layer differently is more applicable than assigning uniform importance. Thus, it is crucial to automatically find the importance of each layer based on the layer's characteristics. We call the importance of each layer its *layer coefficient*.
- **Property 3: Flexible layer participation**
  The layer coefficient can vary across communities. Thus, capturing the optimal layer coefficient *specific to each community* is an important ability since it can distinguish the layer participation in each community. In this case, an algorithm can freely construct a community involved with a sub-

set of layers rather than the entire or common set of available layers.

- **Property 4: Algorithm insensitivity**
  Some approaches are tightly coupled with a specific graph clustering algorithm. This tight coupling may limit the freedom of users to choose a graph clustering algorithm. It is well-known that certain graph clustering algorithms tend to perform particularly well or poorly on certain kinds of graphs [11]. Thus, an ability of applying any clustering algorithms can improve the quality of community detection.
- **Property 5: No layer locality assumption**
  Some approaches find initial communities from a specific layer and then discover final communities by expanding and refining the initial communities on other layers. Those algorithms are regarded to have *locality assumption*. In other words, it is assumed that all hidden communities can be derived from a local region of the layer.
- **Property 6: Independence from the order of layers**
  The results of community detection could be sensitive to the order of processing layers. This limitation typically happens when an algorithm processes layers *sequentially* with a dedicated policy for each layer. In this case, an improper ordering will result in lower-quality results.
- **Property 7: Overlapping layers**
  The communities can be defined in an overlapping way across layers. That is, a vertex can belong to a community C_1 on a certain set of layers but to a community C_2 on another set of layers.

Table 3 shows whether each algorithm supports the seven properties. Our perspective is that more Y's indicate that the algorithm has more powerful and advanced features. Nevertheless, we cannot definitely say that the number of Y's determines the superiority of an algorithm over another. Some algorithm does not need all the properties if it is designed for specific environments. In addition, the performance in terms of efficiency or accuracy is not addressed in Table 3, since an apple-to-apple comparison is not possible owing to the differences in problem settings. Overall, despite of these limitations, we believe that this comparison will give useful insights into various approaches.

Table 3: The comparisons of community detection algorithms for multi-layer graphs.

Algorithm	P1	P2	P3	P4	P5	P6	P7
Li *et al.* [12]	N	N	N	N	N	N	Y
Qi *et al.* [17]	N	Y	N	Y	N	N	Y
Zhou *et al.* [28]	N	Y	N	N	Y	N	N
Xu *et al.* [26]	N	Y	N	N	Y	N	N
Silva *et al.* [21]	N	N	Y	N	Y	N	Y
Ruan *et al.* [18]	N	N	N	Y	Y	Y	N
Tang *et al.* [25]	Y	Y	N	Y	Y	Y	N
Dong *et al.* [6]	Y	Y	N	Y	Y	Y	N
Zeng *et al.* [27]	Y	N	Y	N	Y	Y	Y
Boden *et al.* [1]	Y	N	Y	N	Y	Y	Y

7. FUTURE RESEARCH DIRECTIONS

In this section, we present a few challenging but interesting future research directions.

- **General multi-layer graph applicability**
  Most algorithms covered are only applicable to *pillar* multi-layer graphs. It is definitely true that they are simple but effective to model various real-world situations. However, since a one-to-one correspondence between vertices of different layers is not always guaranteed in the real world, it is more natural to consider an extension of the algorithms into *general* multi-layer graphs.
- **Uncertainty in multi-layer graphs**
  Most studies assume that multi-layer graphs are already cleaned completely. However, in the real world, both vertices and edges could be noisy and ambiguous [23]. For example, in bibliographic datasets, different authors may have the same name. Even worse, information extracted from the real world may not be reliable. Thus, constructing multi-layer graphs with entity resolution and/or trustworthy analysis certainly enhances the quality of the community detection process.
- **Scalability issues**
  In the era of Big Data, the amount of available information grows rapidly. Thus, scalability of both computational time and memory requirement has become a critical issue. Although many researchers are trying to enhance scalability, most studies are being conducted with relatively small datasets because of unsatisfactory scalability. One of feasible solutions is to implement parallel and distributed versions of a community detection algorithm. Another is to use sampling for feature-vector matrices of multi-layer graphs.
- **Temporal analysis**
  Graphs evolve over time, and the communities in graphs also change as time goes by. Thus, understanding and exploiting temporal characteristics are helpful for discovering deep insights about the communities. Although many researchers have studied this problem for single-layer graphs, there is almost no work done for multi-layer graphs. The complexity of modeling the evolution in multi-layer graphs is extremely high since it involves multiple layers and the connections between the multiple layers.

8. CONCLUSIONS

In this paper, we presented a comprehensive understanding of multi-layer graphs and the state-of-the-art community detection algorithms for multi-layer graphs. In recent applications, each entity often engages in multiple relations. Hence, the qualified communities in multi-layer graphs can be discovered by the way of exploiting and fusing all these different aspects of information. We classified community detection algorithms in multi-layer graphs into the six types based on their underlying strategies: cluster expansion, matrix factorization, unified distance, model-based, pattern mining, and graph merging. These algorithms were compared with each other using seven properties. Also, various multi-layer graph datasets used in related studies were summarized for ease of reference. Finally, we tried to provide insights and directions for further research in this domain.

9. ACKNOWLEDGMENTS

This research, "Geospatial Big Data Management, Analysis and Service Platform Technology Development," was supported by the MOLIT (The Ministry of Land, Infrastructure and Transport), Korea, under the national spatial information research program supervised by the KAIA (Korea Agency for Infrastructure Technology Advancement) (15NSIP-B081011-02).

10. REFERENCES

[1] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. In *Proc. 2012 ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 1258–1266, Beijing, China, Aug. 2012.

[2] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, Apr. 2010.

[3] A. Cardillo, J. Gómez-Gardeñes, M. Zanin, M. Romance, D. Papo, F. del Pozo, and S. Boccaletti. Emergence of network features from multiplexity. *Scientific Reports*, 3(1344), Feb. 2013.

[4] F. Celli, F. M. L. D. Lascio, M. Magnani, B. Pacelli, and L. Rossi. Social network data and practices: The case of friendfeed. In *Proc. 3rd Int'l Conf. on Social Computing, Behavioral Modeling, and Prediction*, pages 346–353, Bethesda, Maryland, Mar. 2010.

[5] M. D. Domenico, A. Lima, P. Mougel, and M. Musolesi. The anatomy of a scientific rumor. *Scientific Reports*, 3(2980), Oct. 2013.

[6] X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov. Clustering with multi-layer graphs: A spectral perspective. *IEEE Trans. on Signal Processing*, 60(11):5820–5831, Dec. 2011.

[7] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, Feb. 2010.

[8] N. Kiukkonen, J. Blom, O. Dousse, D. Gatica-Perez, and J. Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. In *Proc. The 7th Int'l Conf. on Pervasive Services*, Berlin, Germany, July 2010.

[9] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. on Knowledge Discovery from Data*, 3(1):1–58, Mar. 2009.

[10] S. Lee, M. Ko, K. Han, and J.-G. Lee. On finding fine-granularity user communities by profile decomposition. In *Proc. 2012 ASONAM Int'l Conf. on Advances in Social Networks Analysis and Mining*, pages 631–639, Istanbul, Turkey, Aug. 2012.

[11] J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In *Proc. 19th Int'l World Wide Web Conf.*, pages 631–640, Raleigh, North Carolina, Apr. 2010.

[12] H. Li, Z. Nie, W.-C. Lee, L. Giles, and J.-R. Wen. Scalable community discovery on textual data with relations. In *Proc. 17th Int'l Conf. on Information and Knowledge Management*, pages 1203–1212, Napa Valley, California, Oct. 2008.

[13] S. Lim, S. Ryu, S. Kwon, K. Jung, and J.-G. Lee. LinkSCAN*: Overlapping community detection using the link-space transformation. In *Proc. 30th Int'l Conf. on Data Engineering*, pages 292–303, Chicago, Illinois, Apr. 2014.

[14] M. Magnani and L. Rossi. The ML-model for multi-layer social networks. In *Proc. 2011 ASONAM Int'l Conf. on Advances in Social Networks Analysis and Mining*, pages 5–12, Kaohsiung City, Taiwan, July 2011.

[15] S. Moon, J.-G. Lee, and M. Kang. Scalable community detection from networks by computing edge betweenness on mapreduce. In *Proc. 2014 Int'l Conf. on Big Data and Smart Computing*, pages 145–148, Bangkok, Thailand, Jan. 2014.

[16] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *Proc. 2005 ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 228–238, Chicago, Illinois, Aug. 2005.

[17] G.-J. Qi, C. C. Aggarwal, and T. Huang. Community detection with edge content in social media networks. In *Proc. 28th Int'l Conf. on Data Engineering*, pages 534–545, Brisbane, Australia, Apr. 2012.

[18] Y. Ruan, D. Fuhry, and S. Parthasarathy. Efficient community detection in large networks using content and links. In *Proc. 22nd Int'l World Wide Web Conf.*, pages 1089–1098, Rio de Janeiro, Brazil, May 2013.

[19] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, Aug. 2007.

[20] A. Silva, W. M. Jr., and M. J. Zaki. Structural correlation pattern mining for large graphs. In *Proc. 8th Workshop on Mining and Learning with Graphs*, pages 119–126, Washington D.C., Aug. 2010.

[21] A. Silva, W. M. Jr., and M. J. Zaki. Mining attribute-structure correlated patterns in large attributed graphs. *Proc. of the VLDB Endowment*, 5(5):466–477, Sept. 2012.

[22] K. Sim, V. Gopalkrishnan, A. Zimek, and G. Cong. A survey on enhanced subspace clustering. *Data Mining and Knowledge Discovery*, 26(2):332–397, Feb. 2013.

[23] Y. Sun and J. Han. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter*, 14(2):20–28, Dec. 2013.

[24] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proc. of the VLDB Endowment*, 4(11):992–1003, Aug. 2011.

[25] W. Tang, Z. Lu, and I. S. Dhillon. Clustering with multiple graphs. In *Proc. 9th Int'l Conf. on Data Mining*, pages 1016–1021, Mianmi, Florida, Dec. 2009.

[26] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *Proc. 2012 ACM SIGMOD Int'l Conf. on Management of Data*, pages 505–516, Indianapolis, Indiana, June 2012.

[27] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *Proc. 2006 ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 797–802, Philadelphia, Pennsylvania, Aug. 2006.

[28] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *Proc. of the VLDB Endowment*, 2(1):718–729, Aug. 2009.