Differential Flattening: A Novel Framework for Community Detection in Multi-Layer Graphs

JUNGEUN KIM, JAE-GIL LEE, and SUNGSU LIM, Korea Advanced Institute of Science and Technology

A *multi-layer graph* consists of multiple layers of weighted graphs, where the multiple layers represent the different aspects of relationships. Considering multiple aspects (i.e., layers) together is essential to achieve a comprehensive and consolidated view. In this article, we propose a novel framework of *differential flattening*, which facilitates the analysis of multi-layer graphs, and apply this framework to community detection. Differential flattening merges multiple graphs into a single graph such that the graph structure with the maximum clustering coefficient is obtained from the single graph. It has two distinct features compared with existing approaches. First, dealing with multiple layers is done *independently* of a specific community detection algorithm, whereas previous approaches rely on a specific algorithm. Thus, any algorithm for a single graph becomes applicable to multi-layer graphs. Second, the contribution of each layer to the single graph is determined *automatically* for the maximum clustering coefficient. Since differential flattening is formulated by an optimization problem, the optimal solution is easily obtained by well-known algorithms such as interior point methods. Extensive experiments were conducted using the Lancichinetti-Fortunato-Radicchi (LFR) benchmark networks as well as the DBLP, 20 Newsgroups, and MIT Reality Mining networks. The results show that our approach of differential flattening leads to discovery of higher-quality communities than baseline approaches and the state-of-the-art algorithms.

CCS Concepts: • **Information systems** → Data mining;

Additional Key Words and Phrases: Clustering coefficient, community detection, differential flattening, multi-layer graphs, social networks

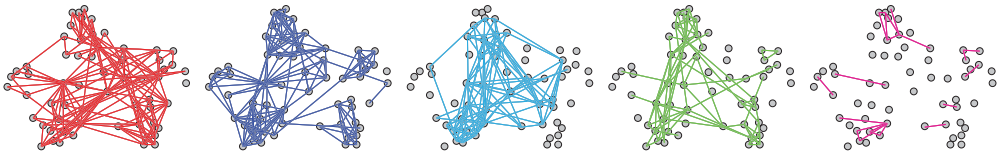
ACM Reference Format:
Jungeun Kim, Jae-Gil Lee, and Sungsu Lim. 2016. Differential flattening: A novel framework for community detection in multi-layer graphs. ACM Trans. Intell. Syst. Technol. 8, 2, Article 27 (October 2016), 23 pages.
DOI: http://dx.doi.org/10.1145/2898362

1. INTRODUCTION

As social contexts have become more complex in recent years, each person is likely to engage in *multiple* roles [Rainie and Wellman 2012]. This phenomenon is accelerated by the proliferation of online social networks such as Facebook and Twitter [Cruz et al. 2013; et al. 2013]. For each role (or identity), people are connected through a specific aspect of relationships, for example, friendships or coworkers. The multiple aspects of relationships can be modeled as multiple interdependent graphs, where each graph

This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. R0101-16-0054, WiseKB: Big data based self-evolving knowledge base and reasoning platform).
Authors' addresses: J. Kim, J.-G. Lee (corresponding author), and S. Lim, Graduate School of Knowledge Service Engineering, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea; emails: {je_kim, jaegil, ssungssu}@kaist.ac.kr.

DOI: http://dx.doi.org/10.1145/2898362



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

Fig. 1. An example of a multi-layer graph called the AUCS data set.

represents an aspect of the relationships. This set of graphs is typically called a *multi-layer graph*.

Figure 1 illustrates an example of a multi-layer graph.[1] The multiple layers represent relationships among 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. All the layers share the common set of vertices, but each layer has a different set of edges. Every single layer contains a piece of meaningful information. Thus, understanding complex social processes requires the analysis of multi-layer graphs.

As the importance of considering multiple aspects is being widely recognized, many researchers have proposed data-mining algorithms for multi-layer graphs, especially for community detection [Kim and Lee 2015]. Tang et al. [2009] and Dong et al. [2012, 2014] proposed algorithms based on spectral clustering for multi-layer graphs. Zeng et al. [2006] and Boden et al. [2012] proposed algorithms that find cliques from multi-layer graphs. They are shown to successfully find meaningful communities from some multi-layer graphs. However, these algorithms are tightly coupled with a specific community detection algorithm, that is, spectral clustering or clique discovery. Therefore, consideration of multiple layers is incorporated into a community detection algorithm.

This tight coupling limits the freedom of users to choose a community detection algorithm. It is known that certain community detection algorithms tend to perform particularly well or particularly poorly on certain kinds of graphs [Leskovec et al. 2010]. Moreover, the properties of communities will vary from algorithm to algorithm. Depending on the particular application and the properties of the graph being analyzed, one might prefer to identify specific types of communities. Thus, to provide more options, various types of community detection algorithms have been proposed for single graphs [Fortunato 2010]. On the other hand, for multi-layer graphs, the types of the available algorithms are quite limited. For this reason, we try to *decouple* a data-mining algorithm from consideration of multiple layers.

Toward this goal, we propose a novel framework of *differential flattening* in this article. Based on the framework, we focus on the problem of community detection from multi-layer graphs. Basically, a single graph is created from a multi-layer graph by differential flattening, and then *ordinary* community detection algorithms are applied to the single graph. Since all the layers are merged into the single graph, the resulting communities should reflect the structures of all the layers. Please note that differential flattening alone does not guarantee optimization of the community quality of a multi-layer graph. Its real benefit is increasing the chances of finding the community detection algorithm that best matches the data set and application at hand.

Differential flattening sets the importance of each layer so they *differ* from each other, which is a natural choice because each aspect of relationships may have different importance in the real world. On the other hand, a naive approach of creating a single graph is to overlay the layers considering the importance of each layer *equally*. We

[1]http://sigsna.net/impact/datasets/.

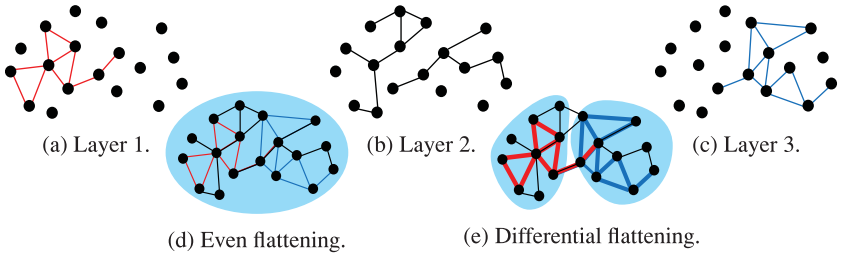


Fig. 2. A motivating example with three layers.

call this naive approach *even flattening* to differentiate it from differential flattening. Since it is reasonable that each layer contributes to the overall community structure at different levels, differential flattening is expected to produce better results than even flattening, as in Example 1.1.

Example 1.1. Suppose that each layer represents a conference in Figures 2(a) to (c). A vertex represents an author and an edge a coauthorship. The first conference focuses on data mining, and the third conference on databases; the second conference accepts both topics. In Figure 2(d), obtained by even flattening, the resulting community contains all the authors. In Figure 2(e), obtained by differential flattening, two communities for each topic are correctly identified. The width of an edge is the edge weight, and the importance of a layer is reflected to the edge weights of the single graph. Here, the first and third layers are considered more important than the second layer since emphasizing them will lead to discovery of *denser* communities. Then, the two communities are easily discovered by a community detection algorithm for *weighted* graphs. Most users will prefer Figures 2(e) to (d). Also, note that these two communities could *not* be precisely detected if we considered each layer *individually*.

This new problem is technically challenging since we need to determine the optimal importance (i) *automatically* and (ii) *before* doing community detection. The latter requirement, which we call the *algorithm insensitivity*, indicates that the quality of the communities discovered from the flattened graph should not be too biased by community detection algorithms. As far as we know, no existing technique satisfies those two requirements. A recent study by Ruan et al. [2013] takes even flattening by default to connect the edges from two graphs. In their study, the weight (importance) of each graph can be adjusted only manually after examining the quality of clustering for various weights.

In summary, the contributions of this article are as follows.

(1) We formalize the framework of *differential flattening* for multi-layer graphs and apply this framework to community detection. The impact of this research is to open the door to the algorithms developed for *single*-layer graphs.
(2) We translate differential flattening to a nonlinear optimization problem and develop an algorithm of finding the optimal solution as well as two techniques for performance improvements.
(3) We demonstrate the effectiveness of differential flattening by conducting extensive experiments on both synthetic and real-world networks.

The rest of the article is organized as follows. Section 2 summarizes state-of-the-art related work. Section 3 formally defines differential flattening. Section 4 presents the algorithm for differential flattening and discusses performance issues. Section 5 reports the results of evaluation. Finally, Section 6 concludes this study.

2. RELATED WORK

In recent years, several community detection (or clustering) algorithms have been extended to process multi-layer graphs [Kim and Lee 2015]. Many research efforts have been devoted to community detection in a *two*-layer graph, which is a special case of a multi-layer graph. This special case appears often in that one graph represents structural similarities while the other represents graph content/attribute similarities. The latter type of similarity is calculated using the contents or attributes associated with a vertex or an edge. Section 2.1 summarizes the approaches for two-layer graphs and Section 2.2 those for general multi-layer graphs, which are more closely related with our approach.

2.1. Two-Layer Graphs

Li et al. [2008] proposed a community discovery algorithm using both relation and attribute information. This algorithm identifies cores through relation analysis and merges the cores using attribute information. Qi et al. [2012] proposed a community detection algorithm by tightly integrating the structural and content aspects of the network using a matrix factorization approach. Zhou et al. [2009] proposed a graph clustering algorithm based on both structural and attribute similarities using a unified distance measure, the Random Walk with Restart (RWR). Xu et al. [2012] proposed a model-based approach, which does not need a distance measure. Silva et al. [2012] proposed a pattern-mining algorithm that finds quasi-cliques sharing common attribute values. Ruan et al. [2013] proposed an approach to combining content and link information for community detection, but it does not determine the weight of a layer automatically.

These algorithms can support only two-layer graphs. Besides, most of them are tightly coupled to a specific graph clustering algorithm and do not consider the contribution of each layer differently.

2.2. General Multi-Layer Graphs

Tang et al. [2009] and Dong et al. [2012, 2014] proposed spectral clustering algorithms for multi-layer graphs. A low-rank matrix factorization approximates a graph by $A \approx P \wedge P^T$, where P is composed of eigenvectors. Linked Matrix Factorization (LMF) [Tang et al. 2009] and Spectral Clustering with Generalized Eigen-Decomposition (SC-GED) [Dong et al. 2012] try to find the common P shared among all the layers. Similarly, Spectral Clustering on Multi-Layer Graphs (SC-ML) [Dong et al. 2014] tries to find the common representation U by modeling each layer as a subspace on the Grassmann manifold and minimizing the distance between U and individual subspaces. Zeng et al. [2006] and Boden et al. [2012] proposed graph clustering algorithms for finding the quasi-cliques that appear on multiple layers more than a given threshold. The main difference between them is how to prune search space: the top-down fashion in Zeng et al. [2006] and the bottom-up fashion in Boden et al. [2012].

These algorithms all depend on a specific community detection algorithm, and the clique-based algorithms do not consider the contribution of each layer differently.

3. DIFFERENTIAL FLATTENING

3.1. Basic Definitions

Table I summarizes the notation used throughout this article. We will omit the superscript (i.e., the layer) to simplify the notation unless we have to mention a specific layer.

A *multi-layer graph* consists of multiple layers, and each *layer* is a weighted graph as defined by Definition 3.1. A few real-world examples are presented in Appendix B.1 (supplementary material, which is available at http://dm.kaist.ac.kr/jaegil/papers/tist16_supp.pdf).

Table I. Summary of the Notation

Notation	Description
$\mathcal{G}_{ML}$	a multi-layer graph
$G^{(l)}$ (or G)	a weighted graph for the l-th layer
G^{flat}	a flattened graph of $\mathcal{G}_{ML}$
v_i	a vertex in V
$e^{(l)}_{i,j}$ (or $e_{i,j}$)	an edge in $E^{(l)}$ between v_i and v_j
$w^{(l)}_{i,j}$ (or $w_{i,j}$)	the weight on the edge $e^{(l)}_{i,j}$
$\widetilde{C}_{v_i}$	a *local* clustering coefficient of v_i
$\widetilde{C}(G^{flat})$	a *global* clustering coefficient of G^{flat}

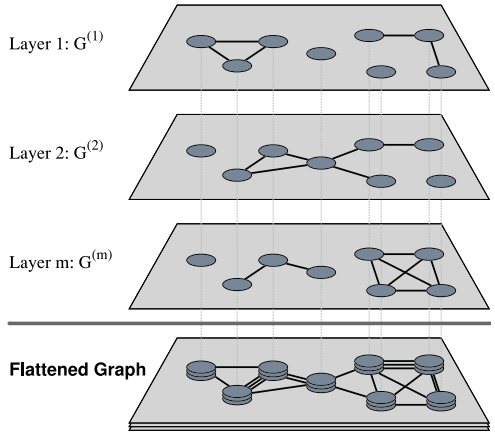


Fig. 3. Flattening of a multi-layer graph.

Note that all the layers share the common set of vertices, whereas each layer has a specific set of edges.[2] The upper part of Figure 3 shows an example of a multi-layer graph.

Definition 3.1. A *multi-layer graph* is a set of weighted graphs $\mathcal{G}_{ML} = \{G^{(1)}, G^{(2)}, \ldots, G^{(m)}\}$, where m is the number of layers. Each *layer* is a weighted graph $G^{(l)} = (V, E^{(l)}, w^{(l)})$, where V is the set of vertices and $E^{(l)}$ the set of edges associated with the weights $w^{(l)}$.

We now formally define the *flattening* of a multi-layer graph in Definition 3.2. For ease of exposition, suppose that $w^{(l)}$ is represented by an *adjacency matrix* $A^{(l)}$, where the element at the ith row and jth column is $w^{(l)}_{i,j}$.

Definition 3.2. The *flattening* of a multi-layer graph is to generate a weighted graph $G^{flat} = (V, E^{flat}, w^{flat})$, where E^{flat} and w^{flat} are defined by Equations (1) and (2). G^{flat} is called a *flattened graph*, and α_l's the *layer coefficients*,

$$E^{flat} = \bigcup_{l=1}^{m} E^{(l)}, \tag{1}$$

$$w^{flat}_{i,j} = [\alpha_1, \alpha_2, \ldots, \alpha_m]\ [w^{(1)}_{i,j}, w^{(2)}_{i,j}, \ldots, w^{(m)}_{i,j}]^T. \tag{2}$$

[2]This model is also called a *pillar multi-network* [Magnani and Rossi 2011].

Equation (2) calculates the dot product between the vector of layer coefficients and that of edge weights.

As in the bottom of Figure 3, a flattened graph is obtained by overlaying all the layers and assigning the weighted sum of $w_{i,j}^{(l)}$'s to each edge of G^{flat}. The layer coefficient is regarded as the importance of the corresponding layer. Here, without losing any generality, we can assume that $\sum_{l=1}^{m} \alpha_l = 1$ and $\alpha_l \in [0, 1]$. This flattening procedure is mathematically represented by a function $F(\mathcal{G}_{ML}, \alpha_1, \ldots, \alpha_m) = G^{flat}$.

3.2. Taxonomy of Flattening

Flattening is first categorized into *even* flattening and *differential* flattening depending on the sameness of α_l's as in Definition 3.3. The latter is more general than the former.

Definition 3.3. *Even* flattening forces all the layer coefficients to be the same, that is, $\alpha_1 = \alpha_2 = \ldots = \alpha_m$. In contrast, *differential* flattening does *not* have such restrictions.

Differential flattening is further categorized into *constrained* differential flattening and *unconstrained* differential flattening depending on the existence of an ordering on α_l's as in Definition 3.4. Constrained flattening is very useful when we have some preferences on the layers. For example, one might want to consider the "KDD" conference more seriously than the "VLDB" conference.

Definition 3.4. *Constrained* differential flattening has ordering constraints on the layer coefficients in terms of their size, for example, $\alpha_2 \leq \alpha_4 \leq \alpha_3$. In contrast, *unconstrained* differential flattening does *not* have such ordering constraints; that is, the layer coefficients are determined purely by an optimality criterion.

3.3. Optimality Criterion

3.3.1. Graph Measures. Before proceeding to an optimality criterion, we first define a *graph measure* in Definition 3.5. Its examples are the edge count, the vertex count, the average degree, the diameter, the clustering coefficient, and so on.

Definition 3.5. A *graph measure* is a function that characterizes a given graph. Mathematically, it is defined by a function $\Phi : \mathbb{G} \rightarrow \mathbb{R}$, which receives a weighted graph and returns a real number.

The *optimality* of differential flattening is now formalized by Definition 3.6, which maximizes a given graph measure for a flattened graph.

Definition 3.6. The differential flattening of a multi-layer graph $\mathcal{G}_{ML}$ is said to be *optimal* when the flattened graph $G^{flat} = F(\mathcal{G}_{ML}, \alpha_1, \ldots, \alpha_m)$ maximizes a given graph measure Φ as in Equation (3),

$$\underset{\alpha_1,\ldots,\alpha_m}{\operatorname{argmax}} \Phi(F(\mathcal{G}_{ML}, \alpha_1, \ldots, \alpha_m)). \tag{3}$$

Thus, our optimization is the selection of the best set of the layer coefficients $\{\alpha_1, \ldots, \alpha_m\}$.

3.3.2. Clustering Coefficients. The graph measure for our problem needs to *predict* the goodness of communities in a graph without actually doing community detection. However, there is no universal definition of a community, and thus many community-goodness measures have been proposed. Common measures are based on density of local edges (*density*), deviation from a random null model (*modularity*), or graph

cuts (*conductance*).[3] Thus, it seems that a single graph measure cannot satisfy these various measures at the same time.

Recently, Gleich and Seshadhri [2012] theoretically proved that the presence of a large global clustering coefficient and a heavy-tailed degree distribution implies the existence of vertex neighborhoods with good *conductance* scores. The *global clustering coefficient* is the ratio of the number of closed triplets to the total number of triplets [Gleich and Seshadhri 2012]. Here, a triplet consists of three vertices that are connected by either two or three undirected edges, and it is closed if three edges exist. When unqualified, the term "clustering coefficient" in this article refers to the global clustering coefficient, which is approximated by the average of the local clustering coefficients of all vertices.

Thus, we employ the clustering coefficient as the graph measure. The two most widely used versions for *weighted* graphs are shown in Equations (4) and (5).

(1) Barrat et al. [2004] defined a local clustering coefficient as Equation (4). Here, k_i is the vertex degree denoting the number of edges connected to a vertex v_i; a_{ij} is 1 if there is an edge between v_i and v_j and 0 otherwise; and $\langle w_i \rangle$ is $\sum_j w_{ij}/k_i$,

$$\widetilde{C}_{v_i,B} = \frac{1}{k_i(k_i-1)} \sum_{j,k} \frac{1}{\langle w_i \rangle} \frac{w_{ij}+w_{ik}}{2} a_{ij}a_{jk}a_{ik}. \tag{4}$$

The contribution of each triangle is weighted by a ratio of the average weight of the two adjacent edges of the triangle to the average weight $\langle w_i \rangle$.

(2) Onnela et al. [2005] defined a local clustering coefficient as Equation (5). Here, $\hat{w}_{ij}$ is $w_{ij}/\max(w)$,

$$\widetilde{C}_{v_i,O} = \frac{1}{k_i(k_i-1)} \sum_{j,k} (\hat{w}_{ij}\hat{w}_{ik}\hat{w}_{jk})^{1/3}. \tag{5}$$

A triangle in which each weight equals $\max(w)$ contributes unity to the sum, whereas a triangle having an edge with a negligible weight will have a negligible contribution.

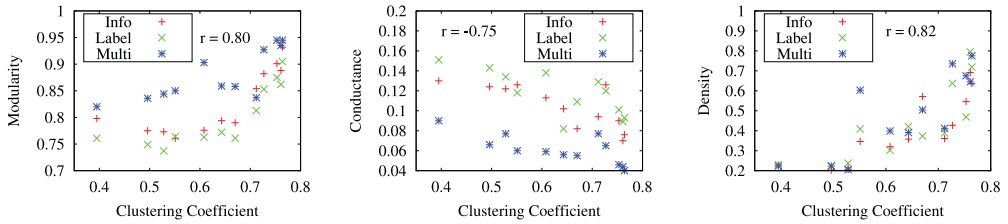
The clustering coefficient is the average of $\widetilde{C}_{v_i,B}$ or $\widetilde{C}_{v_i,O}$ over all v_i's as in Equations (6) and (7). Both $\widetilde{C}_B$ and $\widetilde{C}_O$ are used in our article since each measure has pros and cons [Saramäki et al. 2007]. $\widetilde{C}_O$ can reach the possible maximum value only if the edge weights in each triangle are equal, whereas $\widetilde{C}_B$ can with unequal edge weights in each triangle. Thus, if a user prefers *balanced* weights in triangles (or a community), $\widetilde{C}_O$ is likely to be more suitable than $\widetilde{C}_B$. A subscript "B" or "O" indicates a variation of the clustering coefficient, and it is omitted unless it is needed,

$$\widetilde{C}_B(G^{flat}) = \frac{1}{|V|} \sum_{v_i \in V} \widetilde{C}_{v_i,B}, \tag{6}$$

$$\widetilde{C}_O(G^{flat}) = \frac{1}{|V|} \sum_{v_i \in V} \widetilde{C}_{v_i,O}. \tag{7}$$

We *empirically* confirm the correlations between clustering coefficients and community-goodness measures [Yang and Leskovec 2015]. We generated 12 flattened graphs from the same multi-layer graph by varying the sets of layer coefficients such

[3]Please refer to Fortunato [2010] for the formal definitions.



(a) Modularity. (b) Conductance. (c) Density.

Fig. 4. Correlation between clustering coefficients ($\widetilde{C}_B$) and various clustering qualities.

that those flattened graphs have a rather wide range of clustering coefficients. Then, we run three popular community detection algorithms—Infomap [Rosvall et al. 2009], label propagation [Raghavan et al. 2007], and multi-level modularity [Blondel et al. 2008]—over the graphs. The modularity, conductance, and density of the resulting communities are reported with $\widetilde{C}_B$ in Figure 4. A higher value in modularity and density indicates a better clustering quality, whereas a lower value in conductance indicates a better clustering quality. The measures for the same graph are vertically aligned. The Pearson correlation coefficients are 0.80, -0.75, and 0.82 in Figures 4(a), (b), and (c), respectively. Similar results are obtained also with $\widetilde{C}_O$. That is, modularity, conductance, and density show strong correlations with a clustering coefficient. Therefore, the clustering coefficient can serve as an indicator of "clusterability" and is good to be adopted as the graph measure in this article.

3.4. Problem Setting

In summary, the problem is, for a given multi-layer graph $\mathcal{G}_{ML}$, to find the best set of the layer coefficients $\{\alpha_1, \ldots, \alpha_m\}$ that maximizes the clustering coefficient $\widetilde{C}$ (either $\widetilde{C}_B$ or $\widetilde{C}_O$) of the flattened graph G^{flat}. It is formalized by a nonlinear optimization problem below.

$$\begin{aligned} &\textbf{minimize } -\widetilde{C}(F(\mathcal{G}_{ML}, \alpha_1, \ldots, \alpha_m)) \\ &\textbf{subject to } \textstyle\sum_{l=1}^{m} \alpha_l = 1 \\ &\qquad 0 \leq \alpha_l \leq 1 \text{ for } l = 1, \ldots, m \\ &\qquad [\alpha_i \leq \alpha_j \leq \ldots \leq \alpha_q] \end{aligned}$$

The objective function $-\widetilde{C}(G^{flat})$ is represented by $\alpha_1, \ldots, \alpha_m$ by rewriting $w_{i,j}$'s in Equations (4) or (5) according to Equation (2). The third constraint, which is an ordering of α_l's, is valid only for *constrained* differential flattening.

3.5. Formalization by Diversity Combining

So far, we have discussed that our approach makes the existing clusters denser. However, one might have concern that some *false* communities are induced while making the graph more clusterable. In this section, we formally discuss this issue by adopting a technique widely used in the telecommunication field. Note that this section is *not* meant for introducing an alternative objective function but for showing the merit of the objective function defined in the previous section.

Diversity combining is the technique applied to combine the multiple signals received through independent channels [Brennan 2003]. Since a signal typically experiences fading in the process of signal transmission, a common practice is to establish multiple channels for the signal. By doing so, multiple copies of the signal are available on the

reception side, and these copies are combined in an optimum way to produce a single improved signal.

Differential flattening resembles diversity combining. The real, complex relationships between persons are manifested through multiple graphs, and we are trying to combine these multiple graphs into a single improved graph. In addition, the layers of a multi-layer graph tend to be independent of each other. For example, in Figure 1, a Facebook friendship between two users is independent of the co-authorship between the same two users. Therefore, a layer of a multi-layer graph naturally corresponds to one of the multiple copies of the original signal, and theoretical findings for diversity combining can be applied to differential flattening.

The *signal-to-noise ratio (SNR)* is a measure used for comparing the level of a desired signal to that of background noise. It has also been used to represent the ratio of useful information to false or irrelevant data in social media. In our problem, the useful information indicates the relationships (i.e., edges) that contribute to communities, and a triangle is regarded as a basic unit of a community as supported by the concept of triadic closure [Easley and Kleinberg 2010]. Thus, a set of *signal edges* on an lth layer is first defined by Equation (8), which is a set of the edges on an lth layer that contribute to at least one triangle on a flattened graph. Here, $E_{\triangle}$ denotes a set of the edges on a flattened graph that is part of any triangle. A set of *noise edges* is the rest of $E^{(l)}$,

$$E^{(l)}_{signal} = \{e^{(l)}_{i,j} | E^{(l)} \cap E_{\triangle}\},\ E^{(l)}_{noise} = E^{(l)} - E^{(l)}_{signal} \tag{8}$$

Next, the *signal level* of an lth layer is defined by Equation (9), which is the sum of the weights of the signal edges. The *noise level* is also defined in a similar manner. Then, the *signal power* and *noise power* are defined by the squares of the signal level and noise level, respectively, as in Equation (10),

$$L^{(l)}_{signal} = \sum\nolimits_{e_{i,j} \in E^{(l)}_{signal}} w_{i,j},\ L^{(l)}_{noise} = \sum\nolimits_{e_{u,v} \in E^{(l)}_{noise}} w_{u,v}, \tag{9}$$

$$P^{(l)}_{signal} = (L^{(l)}_{signal})^2,\ P^{(l)}_{noise} = (L^{(l)}_{noise})^2. \tag{10}$$

Last, the *SNR* of the lth layer is defined by the ratio of the signal power to the noise power [Brennan 2003], as in Equation (11),

$$SNR(G^{(l)}) = \frac{P^{(l)}_{signal}}{P^{(l)}_{noise}} = \left(\frac{L^{(l)}_{signal}}{L^{(l)}_{noise}}\right)^2. \tag{11}$$

In Lemma 3.7 and Theorem 3.8, we prove that the SNR of the flattened graph is close to the maximum.

LEMMA 3.7. *If α_l is set to be $L^{(l)}_{signal}$ for $l = 1, \ldots, m$, then $SNR(G^{flat})$ is maximized.*

PROOF. Let $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]$, $\boldsymbol{s} = [L^{(1)}_{signal}, \ldots, L^{(m)}_{signal}]$, and $\boldsymbol{n} = [L^{(1)}_{noise}, \ldots, L^{(m)}_{noise}]$ be the vectors of layer coefficients, signal levels, and noise levels, respectively. Then, the SNR of the flattened graph is derived as Equation (12),

$$SNR(G^{flat}) = \frac{|\boldsymbol{\alpha}\boldsymbol{s}^T|^2}{\mathrm{E}[|\boldsymbol{\alpha}\boldsymbol{n}^{\mathrm{T}}|^2]}. \tag{12}$$

Since we assume the Gaussian noise, the noise power in the denominator of Equation (12) is given by Equation (13), where $N_0 = \mathrm{E}[|\boldsymbol{n}^{\mathrm{T}}\boldsymbol{n}|] = \mathrm{E}[\|\boldsymbol{n}\|^2]$. It is frequently assumed that the noises $\boldsymbol{n}$ have a constant average level [Brennan 2003],

$$\mathrm{E}[|\boldsymbol{\alpha}\boldsymbol{n}^{\mathrm{T}}|^2] = \mathrm{E}[|\boldsymbol{\alpha}\boldsymbol{n}^{\mathrm{T}}\boldsymbol{n}\boldsymbol{\alpha}^{\mathrm{T}}|] = \boldsymbol{\alpha}\mathrm{E}[|\boldsymbol{n}^{\mathrm{T}}\boldsymbol{n}|]\boldsymbol{\alpha}^{\mathrm{T}} = \boldsymbol{\alpha}\mathrm{N}_0\boldsymbol{\alpha}^{\mathrm{T}} = \mathrm{N}_0\|\boldsymbol{\alpha}\|^2. \tag{13}$$

Since constants are not relevant, one can always scale $\boldsymbol{\alpha}$ such that $\|\boldsymbol{\alpha}\| = 1$. The SNR is therefore given by $SNR(G^{flat}) = |\boldsymbol{\alpha}\boldsymbol{s}^T|^2/N_0$. By the Cauchy-Schwarz inequality, as in Equation (14), $SNR(G^{flat})$ becomes a maximum when $\boldsymbol{\alpha}$ is linearly proportional to $\boldsymbol{s}$. In the last equality, N_0 is the sum of the squares of the constant average noise level. The maximum of $SNR(G^{flat})$ is actually the sum of the SNR's from each individual layer,

$$SNR(G^{flat}) = \underbrace{\frac{|\boldsymbol{\alpha}\boldsymbol{s}^T|^2}{N_0} \leq \frac{\|\boldsymbol{\alpha}\|^2\|\boldsymbol{s}\|^2}{N_0}}_{\text{Maximum (equal) when } \boldsymbol{\alpha} \propto \boldsymbol{s}} = \frac{\|\boldsymbol{s}\|^2}{N_0} = \frac{1}{N_0}\sum_{l=1}^{m}(L_{signal}^{(l)})^2 = \sum_{l=1}^{m} SNR(G^{(l)}). \quad (14)$$

This concludes the proof. □

In signal processing, it is common to assume that noises are *independently and identically distributed* in each channel, that is, the uniform power across the frequency band and a normal distribution in the time domain, and the noises can then be modeled as Gaussian noise even though it might not hold in real environments [Brennan 2003]. In our problem, we also take the same noise assumption. First, we assume that the noise edges in each layer are independent since layers tend to be independent. For example, a friendship between *user A* and *user B* in Facebook graph layer is independent of a co-authorship between the same two users in DBLP graph layer. Second, we assume that the noise edges are identically distributed in each layer. The noise edge in our definition does not belong to a triangle (or community) in neither the single layer nor the flattened graph. Thus, noise edges *equally* do not gather together in a certain region.

THEOREM 3.8. *If α_l's for $l = 1, \ldots, m$ are determined by our optimality criterion in Section 3.4, then $SNR(G^{flat})$ is approximately maximized.*

PROOF. By Lemma 3.7, it suffices to show that the optimal $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]$ is linearly proportional to $\boldsymbol{s} = [L_{signal}^{(1)}, \ldots, L_{signal}^{(m)}]$. On the other hand, the optimal $\boldsymbol{\alpha}$ is the set of layer coefficients that maximizes the clustering coefficient of a flattened graph, that is, $\widetilde{C}(G^{flat})$. To simplify the proof of this theorem, we assume that the optimal solution of α_l approximates to $L_{signal}^{(l)}/\sum_j L_{signal}^{(j)}$. This assumption is actually true in that α_l highly correlates with $L_{signal}^{(l)}$ because $L_{signal}^{(l)}$ represents the degree of the contribution from the lth layer to $\widetilde{C}(G^{flat})$. Then, $\alpha_l \approx c \cdot L_{signal}^{(l)}$, where c is a constant, and c is factored out while calculating $SNR(G^{flat})$ in Equation (11). This concludes the proof. □

Theorem 3.8 says that differential flattening boosts signals *without amplifying noises* since it maintains the SNR as high as possible. The implication is that inexistent communities are unlikely to be induced into the flattened graph by depressing noises as well.

4. OPTIMIZATION ALGORITHM

4.1. Convex Functions

In this section, we prove that the optimization problem formulated in Section 3.4 is *strictly quasiconvex* optimization, as stated in Theorem 4.5. Its primary advantage is that if a local minimum exists then it must be a global minimum, as stated in Lemma 4.3. Therefore, we can use any constrained nonlinear optimization solver of finding a local minimum.

Before proceeding, we introduce the definitions of (strictly) quasiconvex and quasiconcave as well as a lemma that guarantees a global minimum with strict

quasiconvexity. If a function f is strictly quasiconvex, then $-f$ is strictly quasiconcave and vice versa.

Definition 4.1. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *quasiconvex* if its domain $dom(f)$ and $S_c = \{x \in dom(f) | f(x) \leq c\}$ for all $c \in \mathbb{R}$ are convex. A function f is called *strictly quasiconvex* if we can replace S_c with $\{x \in dom(f) | f(x) < c\}$.

Definition 4.2. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *quasiconcave* if its domain $dom(f)$ and $S_c = \{x \in dom(f) | f(x) \geq c\}$ for all $c \in \mathbb{R}$ are convex. A function f is called *strictly quasiconcave* if we can replace S_c with $\{x \in dom(f) | f(x) > c\}$.

LEMMA 4.3. *Bazaraa et al. [2006]. Let $f : \mathbb{R}^n \to \mathbb{R}$ be strictly quasiconvex. Then, for the problem to minimize $f(x)$ subject to $x \in S$, where S is a nonempty convex set in $\mathbb{R}^n$, a local minimum solution is also a global minimum.*

PROOF. Refer to Bazaraa et al. [2006] for details. □

In Theorem 4.5, we prove that the two objective functions of our maximization problem, that is, clustering coefficients, are strictly quasiconcave so it is equivalent to a strictly quasiconvex problem, under the condition in which all layer coefficients are positive. Its proof is based on Lemma 4.4. In practice, the condition of Theorem 4.5 holds in our experiments, as shown in Table IV of Section 5.2. Hence, our algorithm for finding a local optimum is guaranteed to find a global optimum *in almost all cases*.

LEMMA 4.4. *Boyd and Vandenberghe [2004] Suppose p is a concave function, q is a convex function, with $p(x) > 0$ and $q(x) > 0$ on a convex set S. Then, the function f defined by $f(x) = p(x)/q(x)$ on S is strictly quasiconcave.*

PROOF. See Appendix A.1. □

THEOREM 4.5. *For any multi-layer graph $\mathcal{G}_{ML}$ with positive layer coefficients, $-\widetilde{C}_O$ and $-\widetilde{C}_B$ are strictly quasiconvex.*

PROOF. We show that the numerators of $\widetilde{C}_O$ and $\widetilde{C}_B$ are concave and the denominators are convex. Then, these functions are strictly quasiconcave by Lemma 4.4. The negation of these strictly quasiconcave functions makes them strictly quasiconvex. See Appendix A.2 for details. □

4.2. Algorithm Descriptions

We use the *interior point method* with the logarithmic barrier function [Waltz et al. 2006], which is blessed with nice convergence results regarding superlinear and quadratic rates of convergence [Forsgren et al. 2002]. Its main idea is to solve a constrained optimization problem via a sequence of subproblems that approximate the original problem.

Each subproblem is formulated as below in our problem. Here, m is the number of layers, q is the number of ordering constraints, $s_1, \ldots, s_{m+q-1} > 0$ are slack variables, and $\mu > 0$ is the barrier parameter. As μ decreases to zero, the optimal objective value

approaches that of the original problem. The logarithmic term enables us to prevent violations of the constraints at all.

$$\begin{aligned}
&\textbf{minimize } -\widetilde{C} - \mu \textstyle\sum_{i=1}^{m+q-1} ln(s_i) \qquad \text{/* } \textit{subproblem} \text{ */}\\
&\textbf{subject to } \textstyle\sum_{l=1}^{m} \alpha_l - 1 = 0\\
&\qquad\qquad -\alpha_i + s_i = 0 \text{ for } i = 1, \ldots, m\\
&\qquad\qquad \alpha_{c_{j-1}} - \alpha_{c_j} + s_{m+j} = 0 \text{ for } j = 1, \ldots, q-1
\end{aligned}$$

Algorithm 1 shows the procedure of finding an optimal solution using the interior point method. In each iteration (Lines 2–7), we solve the subproblem for a fixed value of μ until a certain convergence criterion is satisfied. Then, we decrease the barrier parameter for the next subproblem, repeatedly. In Line 2, we calculate the Lagrangian $\mathcal{L}$ of the subproblem [Boyd and Vandenberghe 2004], which is $-\widetilde{C} + \lambda_0(\sum_{l=1}^{m} \alpha_l - 1) + \sum_{i=1}^{m} \lambda_i(-\alpha_i) + \sum_{j=1}^{q-1} \lambda_{m+j}(\alpha_{c_{j-1}} - \alpha_{c_j})$. Calculating $\widetilde{C}$ in $\mathcal{L}$ involves triangle counting.[4] In Line 3, we apply Newton's method to the Karush-Kuhn-Tucker (KKT) optimality condition to obtain the step-lengths $\Delta\alpha_i$'s, Δs_i's, $\Delta\lambda_i$'s for α_i's, s_i's, λ_i's, respectively. The KKT optimality condition is $\nabla\mathcal{L} = 0$. To implement Newton's step, we need to compute the Hessian $\nabla^2\mathcal{L}$, and it is computed approximately by the solver in many cases. The step-lengths $\Delta\alpha_i$'s, Δs_i's, $\Delta\lambda_i$'s are determined by a descent direction for $\Delta\mathcal{L}$ at α_i's, s_i's, λ_i's.

ALGORITHM 1: Interior Point Method

Require: a multi-layer graph $\mathcal{G}_{ML}$, a graph measure $\widetilde{C}$,
Lagrangian multipliers $\lambda_0, \ldots, \lambda_{m+q-1}$, variables $\alpha_1, \ldots, \alpha_m, s_1, \ldots, s_{m+q-1}$,
tolerance $\epsilon > 0$, barrier parameter $\mu > 0$, $t > 1$;

1: **while** $\sum_{i=1}^{m+q-1} s_i > \epsilon$ **do**
2: Compute Lagrangian $\mathcal{L}$ of the *subproblem*;
3: Apply Newton's method for solving $\nabla\mathcal{L} = 0$;
4: $\alpha_i \leftarrow \alpha_i + \Delta\alpha_i$ for $i = 1, \ldots, m$;
5: $s_i \leftarrow s_i + \Delta s_i$ for $i = 1, \ldots, m + q - 1$;
6: $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$ for $i = 0, \ldots, m + q - 1$;
7: $\mu \leftarrow \mu/t$;
8: **end while**
9: **return** $\alpha_1, \ldots, \alpha_m$; /* an *optimal* solution */

Theorem 4.6 states that the average time complexity of Algorithm 1 is linear if the degree of a graph is much lower than the total number of vertices.

THEOREM 4.6. *Algorithm 1 runs in $O(n\kappa\langle k^2\rangle)$, where n is the number of vertices, κ the number of iterations, and $\langle k^2\rangle$ the average of the squares of the vertex degrees of G^{flat}.*

PROOF. The overall convergence rate is known to be superlinear in the variables and the Lagrangian multipliers [Forsgren et al. 2002]. Thus, the procedure terminates in finite time. For each iteration of Algorithm 1, the number of computations needed for $v_i \in V$ to calculate $\widetilde{C}_{v_i}$ is $O(k_i^2)$ because the number of pairs of the neighbors of v_i is $k_i(k_i - 1)$, where k_i is the degree of v_i in G^{flat}. It holds that $\sum_{i=1}^{n} k_i^2 = n\langle k^2\rangle$. Putting it

[4]Any technique for triangle counting can be used here. In this article, we use *NodeIterator* [Suri and Vassilvitskii 2011], which works by pivoting around each vertex and then checking if there exist the edges in its neighbor vertices.

all together, the time complexity is $O(n\kappa\langle k^2\rangle)$, which is linear to n because $\langle k\rangle \ll n$ in real-world graphs. □

4.3. Performance Issues

We present a heuristic method and a parallelization technique to speed up Algorithm 1.

4.3.1. Initial Point Heuristic. The total number of iterations in Algorithm 1 can be reduced when an initial setting of α_l's is close to an optimal point. In each iteration, the algorithm moves a current estimate of $\boldsymbol{\alpha}^*$ to an updated estimate by estimating the difference between them. In many cases, starting from closer initial points makes the convergence faster.

We propose a simple yet effective heuristic for selecting *initial* layer coefficients in Algorithm 2. We have observed that the signal levels and the layer coefficients are positively correlated in a wide range of problems. In addition, Lemma 3.7 says that $SNR(G^{flat})$, which is closely related to our objective function, is maximized when α_l is proportional to the signal level. Based on this observation and theoretical finding, we assign the initial layer coefficients in the order of the signal levels of individual layers.

ALGORITHM 2: Initial Point Heuristic

Require: a multi-layer graph $\mathcal{G}_{ML}$, constraints on α_l's;
1: **for** $l = 1, \ldots, m$ **do**
2: Compute $L^{(l)}_{signal}$;
3: **end for**
4: **for** $l = 1, \ldots, m$ **do**
5: $\alpha_l \leftarrow L^{(l)}_{signal} / \sum_j L^{(j)}_{signal}$;
6: **end for**
7: **return** $\alpha_1, \ldots, \alpha_m$; /* used as *initial* points */

4.3.2. Parallel Computation. The most time-consuming part of Algorithm 1 is computation of the clustering coefficient (Line 2). We note that the clustering coefficient in Equation (6) or Equation (7) is obtained by taking the average of the *local* clustering coefficients calculated for each vertex. Since the local value is calculated independently of each other, we can parallelize this calculation step by partitioning the set of vertices. While calculating the local clustering coefficients, a thread is assigned to a partition V_i, where $\forall i \neq j$, $V_i \cap V_j = \emptyset$, and $\bigcup V_i = V$, and all the threads run the tasks simultaneously *on a single machine*. Appendix B.5 (supplementary material) includes an illustrative figure.

This technique is easily implemented on a commodity PC equipped with multi-core processors. The number of threads is usually configured to be the number of cores. The adjacency matrix is stored in main memory and shared among all the threads.

5. EXPERIMENTS

We considered the six alternatives below. The first two were used as the baseline approaches, the next two were the variations of differential flattening depending on which definition of a clustering coefficient was used, and the last two were used as the state-of-the-art algorithms. Note that differential flattening in this section refers to *unconstrained* differential flattening.

1. `single max`: taking a *single* layer with the maximum clustering coefficient
2. `even`: even flattening
3. `diff(barrat)`: differential flattening using Equation (6)
4. `diff(onnela)`: differential flattening using Equation (7)

5. `Spectral Clustering on Multi-Layer Graphs (SC-ML)` [Dong et al. 2014]: spectral clustering for multi-layer graphs
6. `Mining Multi-Layered, Attributed Graphs (MiMAG)` [Boden et al. 2012]: clique discovery for multi-layer graphs

After a single graph was generated by one of the first four alternatives, we ran three representative community detection algorithms: Infomap [Rosvall et al. 2009], label propagation [Raghavan et al. 2007], and multi-level modularity [Blondel et al. 2008]. Thus, the *three* sets of results were obtained for a single setting of experiments, that is, one for each community detection algorithm.

Algorithm 1 was implemented using MATLAB R2013b. Our implementation calls the `fmincon` function to run the interior point method. Parallel computation of the clustering coefficient was implemented using the Parallel Computing Toolbox for MATLAB. SC-ML and MiMAG were implemented using MATLAB and Java, respectively. All experiments were done on a 3.4GHz Intel PC with 32GB main memory.

We report the accuracy results using synthetic networks in Section 5.1 and using three real-world networks in Sections 5.2, 5.3, and Appendix B.4 (supplementary material). Then, we report the efficiency results in Section 5.4.

5.1. Synthetic Networks

5.1.1. Data Sets. There is no well-established benchmark for multi-layer graphs since the community detection in multi-layer graphs is a relatively young research area. Thus, we use the networks generated by the Lancichinetti-Fortunato-Radicchi (LFR) benchmark [Lancichinetti and Fortunato 2009], which has been the most widely used model for comparing community detection algorithms. However, since it was designed to create a single graph, we decided to *slice* a single weighted graph into multiple layers so we can still use ground-truth information. This process is reasonable because a single-layer graph does not have clear distinction from a flattening of a multi-layer graph while each layer's explicit information is *hidden*. For example, a DBLP data set has been widely used as a single-layer graph and, at the same time, as a multi-layer graph because it actually consists of multiple layers such as various conferences or journals [Boden et al. 2012]. Thus, we believe that our data generation is *not* biased toward any of the algorithms. In addition, there have been several attempts to separate an arbitrary number of unknown sources from a single mixture based on a set of latent parameters, including *blind source separation* [Mihoko and Eguchi 2002] in the telecommunication field. Our data generation can be considered as a kind of blind source separation in this regard.

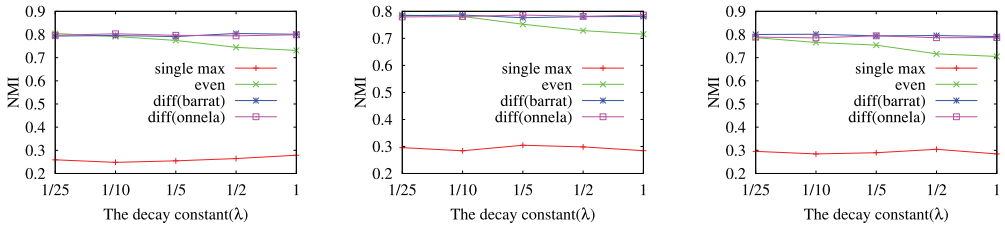
The slice process is described in Algorithm 3, which is self-explanatory. We defined two latent parameters to control hidden information of multiple layers: a *slice coefficient* (σ_l) is the portion of the edge weight that the layer absorbs, and a *multi-edge ratio* (r_{multi}) is the fraction of edges that should exist on multiple layers. Because an edge of the single graph is *not* distributed to all the layers, each layer has a different set of edges and possibly communities. This multi-layer graph is restored to a single graph by differential and even flattening. As the edge weights of the flattened graph get closer to those of the original graph, the resulting communities on the flattened graph will become closer to the ground-truth communities on the original graph. In this way, we measured how well differential flattening uncovers the community structure distributed to and hidden in multiple layers.

We summarize the control parameters in Table II.

5.1.2. Evaluation Metric. The *normalized mutual information (NMI)* is based on the information theory that compares the memberships of two groups [Lancichinetti and Fortunato 2009; Tang and Yang 2014]. Here, one group indicates the results of community detection, and the other group the ground-truth communities. Thus, the NMI

Table II. The Parameters Used for Experiments

Parameter	Description
N	the total number of vertices
m	the number of layers
$\langle k \rangle$	the average degree of a graph
σ_l	the slice coefficient
r_{multi}	the multi-edge ratio
λ	the decay constant of exponential decay



(a) Infomap. (b) Label propagation. (c) Multi-level modularity.

Fig. 5. Effects of layer importance.

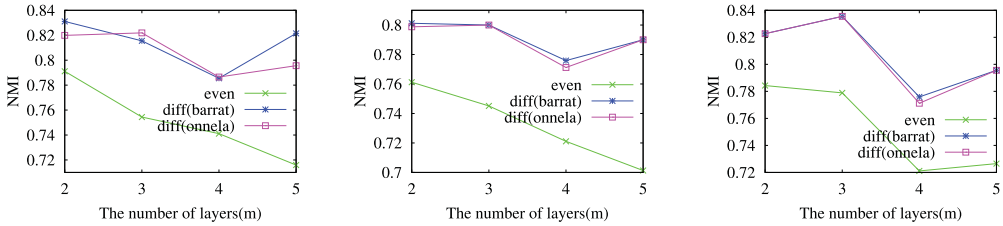
ALGORITHM 3: Graph Slicing (Multiple-Layer Generation)

Require: a weighted graph G, a number of layers m, slice coefficients σ_l's, a multi-edge ratio r_{multi}

for each edge e in G **do**
  $w \leftarrow$ the weight of e;
  Generate a random number $x \in [0, 1]$;
  if $r_{multi} \geq x$ **then**
    Randomly choose *multiple* (about half) layers;
    Assign $w^{(l)}$'s such that $\sum \sigma_l \cdot w^{(l)} = w$;
  else
    Randomly choose a *single* layer;
    Assign $w^{(l)} = w/\sigma_l$ for the selected layer;
  end if
end for
return $\mathcal{G}_{ML}$ generated from G;

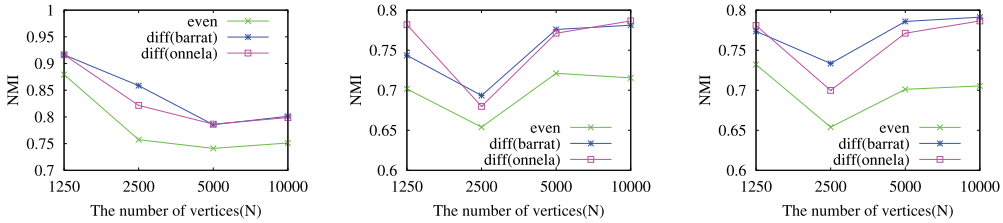
naturally evaluates the quality of community detection. It ranges from 0 to 1, and a higher value represents a better quality. The *adjusted Rand index (ARI)* [Hubert and Arabie 1985] is also adopted as an evaluation metric. However, the results in terms of the NMI and those in terms of the ARI significantly overlapped. More specifically, the Pearson correlation coefficient between the NMI scores and the ARI scores was very high: 0.962 in Section 5.1 and 0.956 in Section 5.3. Thus, we present the results only in terms of the NMI. The formal definitions of the NMI and ARI are presented in Appendix B.2 (supplementary material). Part of the results are given also in terms of the ARI in Appendix B.3 (supplementary material).

5.1.3. Effects of Community Detection Algorithms. In Figures 5–7, a row of three subfigures were generated using the same setting except a community detection algorithm; in each row, the first, second, and third ones represent the results generated by Infomap, label propagation, and multi-level modularity, respectively. The communities identified by different algorithms should differ. Nevertheless, differential flattening is shown to always outperform even flattening, achieving the *algorithm insensitivity*.



(a) Infomap. (b) Label propagation. (c) Multi-level modularity.

Fig. 6. Effects of the number of layers.



(a) Infomap. (b) Label propagation. (c) Multi-level modularity.

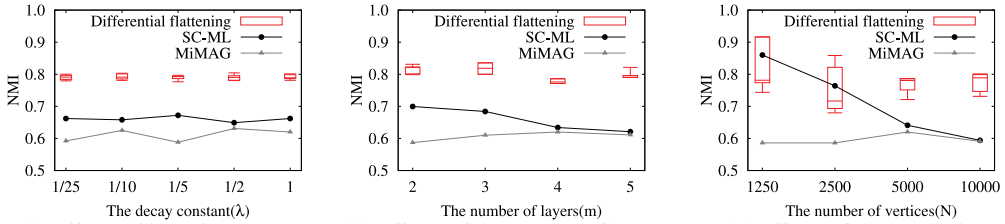
Fig. 7. Effects of the graph size.

5.1.4. Effects of Layer Importance. We first tried various settings for the importance of layers when we sliced an original graph into multiple layers. The exponential decay function $e^{-\lambda x}$ was used to systemically generate various settings. The slice coefficients were configured using the quantity of this function: σ_1 using the quantity at $x = 1$, σ_2 using the quantity at $x = 2$, and so on. As the decay constant λ increases, the quantity vanishes much more rapidly. Thus, the larger λ is, the more skewed the slice coefficients σ_l's are.

Figure 5 shows the effects of the skewness of layer importance. λ was varied from 1/25 to 1 with $N = 5{,}000$, $r_{multi} = 0.4$, and $\langle k \rangle = 5.5$. The performance gap between differential flattening and even flattening gets larger as λ increases, that is, the layer importance becomes more skewed. The layer coefficients of even flattening are actually very close to the slice coefficients when λ is small, but they are not when λ is large. The NMI score of differential flattening is pretty high (over 0.8) for the entire range of λ. Thus, it is shown that differential flattening recovers the overall community structure well regardless of skewness.

In Figures 6 and 7, we set the slice coefficients more realistically without using the decay function. They are $[0.8, 0.2]$, $[0.8, 0.1, 0.1]$, $[0.6, 0.3, 0.05, 0.05]$, and $[0.5, 0.3, 0.1, 0.05, 0.05]$ when the number of layers is 2, 3, 4, and 5, respectively. The decay function is sometimes unrealistic since it monotonically decreases. In addition, `single max` is excluded in subsequent experiments because of its poor performance.

5.1.5. Effects of the Number of Layers. Figure 6 shows the effects of the number of layers. m was varied from 2 to 5 with $N = 5{,}000$, $r_{multi} = 0.4$, and $\langle k \rangle = 5.5$. The performance gap between differential flattening and even flattening increases as the number of layers does, which was mainly caused by the poor performance of even flattening. The layer importance tends to deviate from the uniform distribution as the number of layers increases in our configuration. The consistent dip when m is 4 in Figure 6 was due to our specific setting for the slice coefficients. The important fact is that differential flattening always outperforms even flattening.



(a) Effects of layer importance. (b) Effects of the number of layers. (c) Effects of the graph size.

Fig. 8. Comparison with SC-ML and MiMAG.

5.1.6. Effects of Graph Properties. We tested the effects of two graph properties: the graph size and the layer density. The graph size was controlled by varying N, and the layer density by varying r_{multi}. The average density of layers will increase as r_{multi} does. Figure 7 shows the effects of the graph size. N was varied from 1,250 to 10,000 with $m = 4$, $r_{multi} = 0.4$, and $\langle k \rangle = 5.5$. The performance gap between differential flattening and even flattening is constantly significant regardless of the graph size. The same conclusion can be drawn regarding the layer density. These results prove that our framework is robust to the structure of a graph.

5.1.7. Comparison with SC-ML and MiMAG. We repeated the experiments for Figures 5, 6, and 7 using SC-ML [Dong et al. 2014] and MiMAG [Boden et al. 2012], and the results are reported in Figures 8(a), (b), and (c), respectively. For SC-ML, the regularization parameter λ was set to be a default value 0.5, and the number of communities k was set to be the number of ground-truth communities to favor SC-ML. For MiMAG, the density parameter γ and the redundancy parameter r were set to be the default values 0.5 and 0.25, respectively. The vertices not in the results were excluded from calculating the NMI score for MiMAG because it did not assign cluster labels to all vertices. For each configuration, only one NMI score is produced by SC-ML and MiMAG since they are tightly coupled with *spectral clustering* and *clique discovery*, respectively, whereas six scores are produced by differential flattening because of *two* variations of clustering coefficients×*three* community detection algorithms. These *six* scores were represented by a box plot and may have a wide spectrum because of the different characteristics of community detection algorithms.

In general, the accuracy of SC-ML was much worse than that of differential flattening and decreased as the number of layers increased (Figure 8(b)) or the graph size increased (Figure 8(c)). In Figure 8(c), even though SC-ML outperformed differential flattening in *some* cases when $N < 5000$, its accuracy was still worse than the *best* accuracy of differential flattening. In addition, the accuracy of MiMAG was consistently poor in all configurations.

We contend that this poor accuracy of SC-ML and MiMAG was mainly caused by the deficiencies in the underlying algorithm. With regard to SC-ML, it is known that spectral clustering often fails to discover proper communities when they have different scales of size or density [Nadler and Galun 2006]. In Figure 8(c), in fact, the size of the ground-truth communities became more diverse as N increased. When N was 1,000, 2,500, 5,000, and 10,000, the standard deviations of the size were 12.25, 24.79, 142.77, and 308.45, respectively. With regard to MiMAG, a considerable portion of the communities discovered were too small compared with the ground-truth communities, because a clique is inherently a very strict structural property [Fortunato 2010]. Therefore, being able to freely choose a community detection algorithm indeed shows the necessity of *decoupling* data-mining algorithms from consideration of multiple layers. Domain

Table III. The Characteristics of Conferences

Area	Conference	# of Vertices	# of Edges
IR	SIGIR	2,203	4,171
DB	VLDB	2,951	8,114
DB	SIGMOD	2,615	7,766
DB	ICDE	3,212	6,983
DM	KDD	1,913	3,580
DM	ICDM	1,458	2,378
Inter	WWW	1,932	3,478
Inter	CIKM	2,179	3,560
Inter	ICME	1,950	3,119
-	All	14,458	37,068

Table IV. The Seven Combinations of Conference Areas

Case	Combinations	Conferences	Layer Coefficients (Barrat)	Layer Coefficients (Onnela)
1	IR+Inter*2	SIGIR/WWW/CIKM	0.487/0.373/0.139	0.866/0.067/0.067
2	DB+Inter*2	VLDB/WWW/ICME	0.522/0.398/0.08	0.848/0.083/0.069
3	DM+Inter*2	KDD/WWW/CIKM	0.393/0.384/0.223	0.866/0.067/0.067
4	IR+DB+Inter*2	SIGIR/ICDE/CIKM/WWW	0.3/0.282/0.183/0.235	0.44/0.35/0.21/0.05
5	IR+DM+Inter*2	SIGIR/KDD/CIKM/WWW	0.3/0.26/0.13/0.309	0.298/0.607/0.05/0.093
6	DB+DM+Inter*2	SIGMOD/ICDM/ICME/CIKM	0.46/0.232/0.129/0.18	0.834/0.166/0.05/0.05
7	IR+DB+DM+Inter	SIGIR/ICDE/KDD/CIKM	0.339/0.382/0.198/0.081	0.127/0.51/0.05/0.364

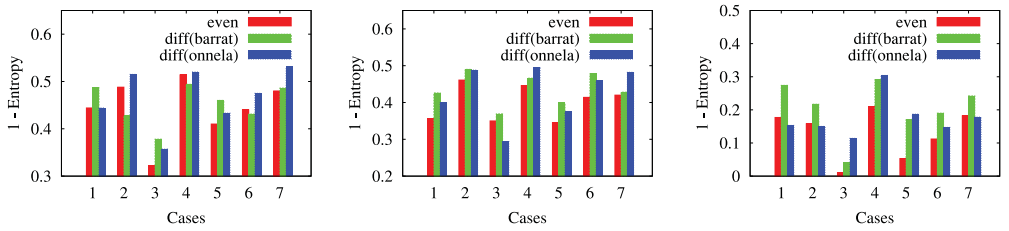
experts may investigate the results of each algorithm and determine which algorithm to use for their analysis.

5.2. DBLP Networks

5.2.1. Data Sets. We used a DBLP data set consisting of 14,458 vertices and 37,068 edges for the articles in the years from 1980 to 2006. A vertex represents an author, and an edge a coauthorship. A venue—either a conference or a journal—was regarded as a layer as in other studies [Boden et al. 2012], and it serves as a layer very well because each venue has a distinct set of topics of interest. For example, a SIGMOD article coauthored by Michael and Jim creates an edge between them on the SIGMOD layer, and a KDD article coauthored by David and Robert creates an edge between them on the KDD layer. We categorized some popular conferences into one of four areas: information retrieval (IR), databases (DB), data mining (DM), and interdisciplinarity (Inter), as in Table III. WWW, CIKM, and ICME have a stronger inclination to accept a wide range of topics than others. Then, we made seven combinations of the four areas to generate various multi-layer graphs, as in Table IV.

5.2.2. Evaluation Metrics. We need to know who are *mainly* interested in IR, DB, or DM. To this end, we consulted the program committee of CIKM since it manages indeed well-organized tracks for each area.[5] Three sets of authors—616 authors in total—were collected from the members of the program committee in the years from 2004 to 2009, and they were used to label each author with his/her main interest. Then, for each community of authors, we calculated the *entropy* of the authors' main interests. The entropy was used instead of the NMI since the community labels were available only for a subset of the authors. A lower entropy value indicates that more authors in a community share the common interest, thereby verifying a better clustering. The

[5]KM (knowledge management) is equivalent to DM.



(a) Infomap. (b) Label propagation. (c) Multi-level modularity.

Fig. 9. Community homogeneity in the DBLP data set.

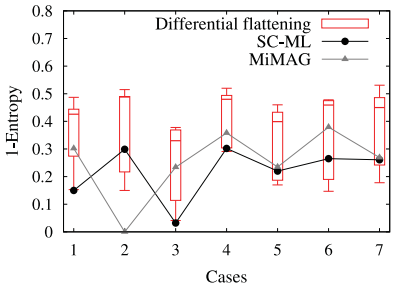


Fig. 10. Comparison with SC-ML and MiMAG.

authors who are not in the ground truth were excluded in calculating the entropy. Last, we calculated the weighted arithmetic mean of the entropies obtained from individual communities, where a weight is the number of the authors in a community. The formal definition of this metric is presented in Appendix B.2 (supplementary material).

5.2.3. Community Homogeneity. Figure 9 shows the entropy of differential flattening and even flattening. Let *entropy* be a weighted arithmetic mean of entropies. The y-axis displays $1 - entropy$ so a better one has a higher value. Both `diff (barrat)` and `diff(onnela)` outperform even flattening in many cases, and at least one of them always outperforms even flattening. Since the p-values are less than a typical significance level 0.05, the differences between differential flattening and even flattening are statistically significant.

One of the main reasons for this lower entropy is illustrated in Example 1.1, which is our motivating example. The authors of the same research interest are more closely knitted in "focused" conferences such as SIGIR, VLDB, and KDD than in "interdisciplinary" conferences such as WWW, CIKM, and ICME. Table IV shows the optimal layer coefficients obtained by differential flattening. In general, those of the focused conferences are higher than those of the interdisciplinary conferences according to our expectations.[6] Therefore, we conclude that differential flattening results in more homogeneous communities than even flattening.

5.2.4. Comparison with SC-ML and MiMAG. We repeated the experiments for Figure 9 using SC-ML and MiMAG, and the results are reported in Figure 10. All parameter values of SC-ML and MiMAG in this section are identical to those in Section 5.1.7. In most cases, SC-ML and MiMAG showed better performance than the worst accuracy of differential flattening. However, not only the highest accuracy but also the average

[6]This trend is almost always true, but it might not for some combinations of layers. For example, in Case 7 of Table IV, an exception happened probably because the triangles on the KDD layer highly overlap those on the CIKM layer.

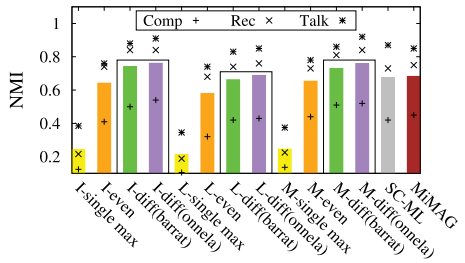


Fig. 11. Accuracy for the 20 Newsgroups data set.

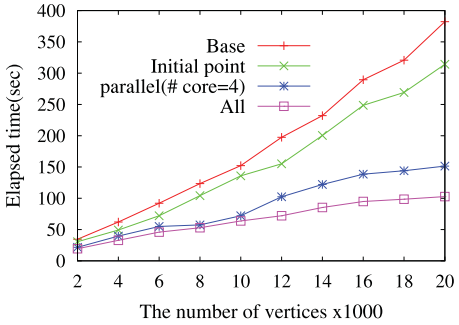


Fig. 12. Elapsed time (LFR benchmark).

accuracy of differential flattening outperformed both SC-ML and MiMAG in all cases. The major reason for the superiority of differential flattening is similar to the reason discussed in previous sections. For the comparison between the two external methods, MiMAG showed a little higher accuracy than SC-ML in most cases since MiMAG produced relatively smaller but denser communities which contributed to lower entropy.

5.3. 20 Newsgroups Networks

5.3.1. Data Sets. We evaluated differential flattening using the 20 Newsgroups data set [Ni et al. 2015][7]. We selected four news groups from each of three categories—Comp, Rec, and Talk—and derived a multi-layer graph for each category. That is, *three* multi-layer graphs were generated. A multi-layer graph consisted of *five* layers, each with *four* communities corresponding to news groups. For each layer, we added *400* vertices by randomly sampling 100 documents from each of four news groups. A vertex (i.e., document) in one layer was randomly mapped to a vertex in another layer if they were sampled from the same news group, and these two vertices on *different* layers were considered to have the same identifier. The similarity between two documents was computed by cosine similarity, and an edge between two vertices on the *same* layer was created if cosine similarity was above a certain threshold value.

5.3.2. Evaluation Metrics. We adopted predefined news groups as the ground truth and the NMI as an evaluation metric. The ground truth of each category had the four communities as follows: graphics, os.ms-windows.misc, sys.ibm.pc.hardware, and sys.mac.pc.hardware in **Comp**; autos, motorcycles, sport.baseball, and sport.hockey in **Rec**; and politics.guns, politics.mideast, politics.misc, and religion.misc in **Talk**.

5.3.3. Accuracy Result. Figure 11 shows the NMI scores generated by all alternatives. The scores are presented individually for the three multi-layer graphs, and a bar indicates the *average* of the three scores. The first letter—**I**, **L**, or **M**—in the algorithm name denotes the community detection algorithm combined with our approach. In general, the accuracy of differential flattening was better than that of all other alternatives. Differential flattening outperformed SC-ML by up to 29% and MiMAG by up to 20%. In addition, we observed that the Comp multi-layer graph contained much more inter-community edges than the Rec and Talk multi-layer graphs. The proportions of such edges were 0.35, 0.12, and 0.08 in the Comp, Rec, and Talk multi-layer graphs, respectively. Since community detection becomes harder with more inter-community edges, this high proportion explains unsatisfactory accuracy for the Comp multi-layer graph in all alternatives.

[7]http://qwone.com/~jason/20Newsgroups/.

5.4. Running Time

Figure 12 shows the elapsed time of our approach with two speed improvements. Using the LFR benchmark as in Section 5.1, N was varied from 1,000 to 20,000 with $m = 4$, $r_{multi} = 0.4$, and $\langle k \rangle = 5.5$. `Base` is the default implementation using the sparse matrix and the compressed row storage. `Initial Point` is the initial point heuristic (Section 4.3.1) built on the default implementation. `Parallel` is the parallelization technique (Section 4.3.2) built on the default implementation. Here, four threads were created since our PC has a quad-core processor. `All` has both improvements. First, the elapsed time of `Base` increases *linearly* as the number of vertices does. This linear scalability conforms to Theorem 4.6. As for the effects of the two improvements, compared with `Base`, the elapsed time was improved by up to 27% with `Initial Point`, by up to 245% with `Parallel`, and by up to 273% with `All`. `Parallel` was more effective than `Initial Point`. Overall, `All` is shown to process a dense graph with 20,000 vertices and four layers within 2 minutes on a commodity PC. Expecting the same trend to continue, `All` will be able to handle 100,000 vertices within 10 minutes, as long as main memory is enough to hold the data set. This time can be further reduced if more CPU cores are available.

Calculating the clustering coefficient involves triangle counting (or listing). Since it is one of the core operations in social network analysis, many techniques have been developed for efficient triangle counting [Suri and Vassilvitskii 2011]. These techniques are orthogonal to our work and can be adopted in order to further improve scalability.

6. CONCLUSIONS

In this article, we proposed a novel framework of *differential flattening*. It enables us to optimally merge multiple layers into a single layer with considering the importance of each layer differently. Finding the optimal merger is translated to solving a nonlinear optimization problem. Its prominent advantage is to open the door to the algorithms for a *single*-layer graph so they can be applied to a *multi*-layer graph. This framework was first adopted to the problem of community detection.

We conducted extensive experiments using synthetic and real-world networks. First, the results for the LFR benchmark networks, the 20 Newsgroups networks, and the Reality Mining networks showed that differential flattening led to higher accuracy than even flattening and the state-of-the-art algorithm. Second, the results for the DBLP networks demonstrated that differential flattening put more emphasis on "focused" conferences rather than on "interdisciplinary" conferences, resulting in more homogeneous communities of researchers. Overall, we believe that we have provided an interesting direction of the analysis of multi-layer graphs. Our future work includes evaluating the effectiveness of the ordering constraints and applying the framework to other data-mining problems such as link prediction and influence propagation.

APPENDIX A

A.1. Proof of Lemma 4.4

Let $S_c = \{x \in S | f(x) > c\}$ be a superlevel set for $c \in \mathbb{R}$. If $c \leq 0$, then $S_c = S$ so S_c is convex. Otherwise, $c > 0$ and $p(x) - cq(x)$ is a concave function since p and $-q$ are concave. $S_c = \{x \in S | f(x) > c\} = \{x \in S | p(x) - cq(x) > 0\}$ is convex since a superlevel set of a concave set is convex. Thus, S_c is convex for all $c \in \mathbb{R}$, and f is strictly quasiconcave by Definition 4.2. □

A.2. Proof of Theorem 4.5

It suffices to show that $-\widetilde{C}_{v_i,O}$ and $-\widetilde{C}_{v_i,B}$ are strictly quasiconvex for any vertex v_i. Let $\boldsymbol{\alpha} = (\alpha_l)_{l=1,\ldots,m}$ be a vector consisting of a set of layer coefficients. For all $v_i, v_j \in V$, let

$\mathbf{w}_{ij} = (w_{ij}^{(l)})_{l=1,\ldots,m}$. We define $k_i^{(\boldsymbol{\alpha})}$ by the number of edges connected to v_i in G^{flat} with a vector $\boldsymbol{\alpha}$. Assume that all layer coefficients are positive, that is, all coordinates of $\boldsymbol{\alpha}$ are non-zero. Then, the number of adjacent edges is not affected by $\boldsymbol{\alpha}$ because two vertices are adjacent in the flattened graph if and only if they are adjacent in at least one layer, though the weights of the adjacent edges are affected by $\boldsymbol{\alpha}$. Hence, $k_i^{(\boldsymbol{\alpha})}$ is the same for every α, and we call it k_i.

Proof for $-\widetilde{C}_{v_i,O}$: Equation (5) is rewritten as Equation (15),

$$\widetilde{C}_{v_i,O}(F(\mathcal{G}_{ML}, \boldsymbol{\alpha})) = \frac{1}{k_i(k_i-1)\max(\boldsymbol{\alpha}\cdot\mathbf{w})} \sum_{j,k}((\boldsymbol{\alpha}\cdot\mathbf{w}_{ij})\cdot(\boldsymbol{\alpha}\cdot\mathbf{w}_{jk})\cdot(\boldsymbol{\alpha}\cdot\mathbf{w}_{ik}))^{1/3}. \tag{15}$$

Remark that k_i is a constant for a fixed v_i. Each $\boldsymbol{\alpha}\cdot\mathbf{w}_{ij}$ is a linear function of $\boldsymbol{\alpha}$ that is convex, and the pointwise maximum of convex functions is known to be convex [Boyd and Vandenberghe 2004]. Hence, the denominator is convex. Also, we can easily check that the numerator is concave since the geometric mean is known to be concave. By Lemma 4.4, we obtain that $\widetilde{C}_{v_i,O}$ is strictly quasiconcave for all $v_i \in V$ and $-\widetilde{C}_{v_i,O}$ is strictly quasiconvex.

Proof for $-\widetilde{C}_{v_i,B}$: Equation (4) is rewritten as Equation (16),

$$\widetilde{C}_{v_i,B}(F(\mathcal{G}_{ML}, \boldsymbol{\alpha})) = \frac{1}{2(k_i-1)\sum_{j'}\boldsymbol{\alpha}\cdot\mathbf{w}_{ij'}} \sum_{j,k}(\boldsymbol{\alpha}\cdot\mathbf{w}_{ij}+\boldsymbol{\alpha}\cdot\mathbf{w}_{ik})a_{ij}^{(\alpha)}a_{jk}^{(\alpha)}a_{ik}^{(\alpha)}. \tag{16}$$

Note that a linear function is both convex and concave [Boyd and Vandenberghe 2004]. The denominator and numerator of $\widetilde{C}_{v_i,B}$ are the linear functions of $\boldsymbol{\alpha}$, so they are both convex and concave. By Lemma 4.4, we also conclude that $-\widetilde{C}_{v_i,B}$ is strictly quasiconvex. □

REFERENCES

A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. 2004. The architecture of complex weighted networks. *Proc. Natl. Acad. Sci. U.S.A.* 101, 11 (2004), 3747–3752.

Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. 2006. *Nonlinear Programming: Theory and Algorithms* (3 ed.). Wiley.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theor. Exp.* 2008, 10 (2008), P10008.

Brigitte Boden, Stephan Günnemann, Holger Hoffmann, and Thomas Seidl. 2012. Mining coherent subgraphs in multi-layer graphs with edge labels. In *KDD*. 1258–1266.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

D. G. Brennan. 2003. Linear diversity combining techniques. *Proc. IEEE* 91, 2 (2003), 331–356.

Juan David Cruz, Cécile Bothorel, and François Poulet. 2013. Community detection and visualization in social networks: Integrating structural and semantic information. *ACM Trans. Intell. Syst. Technol.* 5, 1 (2013), 11.

Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. 2012. Clustering with multi-layer graphs: A spectral perspective. *IEEE Trans. Sign. Process.* 60, 11 (2012), 5820–5831.

Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. 2014. Clustering on multi-layer graphs via subspace analysis on Grassmann manifolds. *IEEE Trans. Sign. Process.* 62, 4 (2014), 905–918.

David Easley and Jon Kleinberg. 2010. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press.

Michael Fire et al. 2013. Computationally efficient link prediction in a variety of social networks. *ACM Trans. Intell. Syst. Technol.* 5, 1 (2013), 10.

Anders Forsgren, Philip E. Gill, and Margaret H. Wright. 2002. Interior methods for nonlinear optimization. *SIAM Rev.* 44, 4 (2002), 525–597.

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

David F. Gleich and C. Seshadhri. 2012. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *KDD*. 597–605.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *J. Classif.* 2, 1 (1985), 193–218.

Jungeun Kim and Jae-Gil Lee. 2015. Community detection in multi-layer graphs: A survey. *ACM SIGMOD Record* 44, 3 (2015), 37–48.

Andrea Lancichinetti and Santo Fortunato. 2009. Community detection algorithms: A comparative analysis. *Phys. Rev. E* 80, 5 (2009), 056117.

Jure Leskovec, Kevin J. Lang, and Michael W. Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *WWW*. 631–640.

Huajing Li, Zaiqing Nie, Wang-Chien Lee, Lee Giles, and Ji-Rong Wen. 2008. Scalable community discovery on textual data with relations. In *CIKM*. 1203–1212.

Matteo Magnani and Luca Rossi. 2011. The ML-model for multi-layer social networks. In *ASONAM*. 5–12.

Minami Mihoko and Shinto Eguchi. 2002. Robust blind source separation by beta divergence. *Neur. Comput.* 14, 8 (2002), 1859–1886.

Boaz Nadler and Meirav Galun. 2006. Fundamental limitations of spectral clustering. In *Advances in Neural Information Processing Systems*. 1017–1024.

Jingchao Ni, Hanghang Tong, Wei Fan, and Xiang Zhang. 2015. Flexible and robust multi-network clustering. In *KDD*. 835–844.

Jukka-Pekka Onnela, Jari Saramäki, János Kertész, and Kimmo Kaski. 2005. Intensity and coherence of motifs in weighted complex networks. *Phys. Rev. E* 71, 6 (2005), 065103.

Guo-Jun Qi, Charu C. Aggarwal, and Thomas Huang. 2012. Community detection with edge content in social media networks. In *ICDE*. 534–545.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* 76, 3 (2007), 036106.

Lee Rainie and Barry Wellman. 2012. *Networked: The New Social Operating System*. The MIT Press.

Martin Rosvall, Daniel Axelsson, and Carl T. Bergstrom. 2009. The map equation. *Eur. Phys. J. Spec. Top.* 178, 1 (2009), 13–23.

Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. 2013. Efficient community detection in large networks using content and links. In *WWW*. 1089–1098.

Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo Kaski, and Janos Kertesz. 2007. Generalizations of the clustering coefficient to weighted complex networks. *Phys. Rev. E* 75, 2 (2007), 027105.

Arlei Silva, Wagner Meira Jr., and Mohammed J. Zaki. 2012. Mining attribute-structure correlated patterns in large attributed graphs. *Proc. VLDB* 5, 5 (2012), 466–477.

Siddharth Suri and Sergei Vassilvitskii. 2011. Counting triangles and the curse of the last reducer. In *WWW*. 607–614.

Wei Tang, Zhengdong Lu, and Inderjit S. Dhillon. 2009. Clustering with multiple graphs. In *ICDM*. 1016–1021.

Xuning Tang and Christopher C. Yang. 2014. Detecting social media hidden communities using dynamic stochastic blockmodel with temporal Dirichlet process. *ACM Trans. Intell. Syst. Technol.* 5, 2 (2014), 36.

R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban. 2006. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Math. Program.* 107, 3 (2006), 391–408.

Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2012. A model-based approach to attributed graph clustering. In *SIGMOD*. 505–516.

Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowl. Inform. Syst.* 42, 1 (2015), 181–213.

Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. 2006. Coherent closed quasi-clique discovery from large dense graph databases. In *KDD*. 797–802.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proc. VLDB* 2, 1 (2009), 718–729.

Received November 2015; accepted February 2016