Motif-based embedding for graph clustering

J. Stat. Mech. (2016) 123401

(http://iopscience.iop.org/1742-5468/2016/12/123401)

Journal of Statistical Mechanics: Theory and Experiment
An IOP and SISSA journal

PAPER: Interdisciplinary statistical mechanics

Motif-based embedding for graph clustering

Sungsu Lim and Jae-Gil Lee[1]

Graduate School of Knowledge Service Engineering, KAIST, Daejeon 34141, Korea
E-mail: ssungssu@kaist.ac.kr and jaegil@kaist.ac.kr

Received 24 March 2016, revised 6 August 2016
Accepted for publication 2 September 2016
Published 1 December 2016

Online at stacks.iop.org/JSTAT/2016/123401
doi:10.1088/1742-5468/2016/12/123401

Abstract. Community detection in complex networks is a fundamental problem that has been extensively studied owing to its wide range of applications. However, because community detection methods typically rely on the *relations* between vertices in networks, they may fail to discover *higher-order* graph substructures, called the *network motifs*. In this paper, we propose a novel embedding method for graph clustering that considers higher-order relationships involving multiple vertices. We show that our embedding method, which we call *motif-based embedding*, is more effective in detecting communities than existing graph embedding methods, spectral embedding and force-directed embedding, both theoretically and experimentally.

Keywords: analysis of algorithms, clustering techniques, random graphs, networks

J. Stat. Mech. (2016) 123401

[1] Author to whom any correspondence should be addressed.

1742-5468/16/123401+18$33.00

Contents

1. Introduction

In a highly networked society, large amounts of network data are being generated everyday. The importance of massive network analysis is increasing rapidly. In particular, *graph clustering*, also known as *community detection* in networks, is one of the most well-established problems in network analysis [6, 12, 37]. It is a procedure of finding the community structure, with many edges joining vertices in the same community and relatively few edges joining vertices of different communities. Community detection has been applied to many domains such as biological networks, social networks, information networks, and so on. Due to its large applicability, many graph clustering methods based on distinct principles have been proposed to identify hierarchical organization [34], overlapping community structure [24], multi-relational community structure [16], and so on.

As a methodology of graph clustering, we decided to use graph embedding, which is theoretically pleasing and has been widely used for this purpose [23, 25, 29]. It is

doi:10.1088/1742-5468/2016/12/123401

J. Stat. Mech. (2016) 123401

known that graph embedding finds a low-dimensional mapping that preserves the connectivity structure of a given graph [39]. Using this property of embedding, clustering algorithms based on embedding detect clusters in a low-dimensional space. These approaches first map vertices of a given graph to points in a low-dimensional space and then group the points that are closer from each other into clusters. The best-known method in this category is spectral clustering [27, 40] using nonlinear dimensionality reduction. Nonlinear dimensionality reduction methods were developed originally for embedding *vectorial data*, but recently have also been extensively used for *graph data* [25]. However, it is known that these methods often fail to find good low-dimensional embeddings for graphs since the vertices with high degrees are crowded in the middle while those with low degrees are scattered in the output space [43]. On the other hand, in order to *visualize* graphs, force-directed embedding is being widely used [39]. It finds an equilibrium of vertex positions where there exist *attractive* forces between adjacent vertices and *repulsive* forces between every two vertices. This method can resolve the limitation of spectral clustering using degree-weighted repulsion so that the high-degree vertices can be placed separately. In this respect, we follow the framework of force-directed embedding for graph clustering.

A notable trend in graph clustering is to consider *higher-order* graph substructures, called the *network motifs* or *graphlets* [1, 4, 5]. The order of a graph substructure is determined by the number of vertices required to express the substructure. Previous studies on graph clustering have mainly focused on considering pairwise relationships between vertices, i.e. second-order substructures. It is known that using higher-order relationships that involve multiple vertices can improve the understanding of the underlying graph structure [4, 5]. For instance, a triangle, which is a third-order relationship between vertices, is regarded as a fundamental substructure for understanding social networks [19] and their communities [35]. The *motif-based weighting* method can be used to reflect the effect of motifs on each pair of vertices [4, 5]. We illustrate the motif-based weighting method with an example in figure 1. The left graph is an original undirected graph with two clusters. Here, the weight between two vertices is defined as the sum of the number of edges and motifs in which the two vertices are contained together, where we set a triangle to be a motif of interest. With the resulting weights, the right graph more readily shows the community structure in the perspective of inter- and intra-cluster weights. Hence, we adopt the motif-based weighting method. Putting together force-directed embedding and motif-based weighting, our method increases the attraction within motifs so that their positions are located in similar positions and grouped easily.

Some recent studies have started to pay attention to identify clusters using network motifs. For graph clustering *without embedding*, there have been several attempts to generalize the evaluation measures for graph clustering by defining inter- and intra-cluster connections with respect to motifs rather than edges, e.g. the motif-based modularity [1], the motif-based cut [17], and the motif-based conductance [4, 5]. For graph clustering *with embedding*, a tensor spectral clustering algorithm has been recently proposed, which can be applied to discover higher-order substructures of particular interest [4]. However, to the best of our knowledge, no previous work has addressed the problem of *force-directed embedding with motifs*.

In this paper, we incorporate motifs into force-directed embedding in support of graph clustering. In summary, the contributions of this paper are as follows.

J. Stat. Mech. (2016) 123401

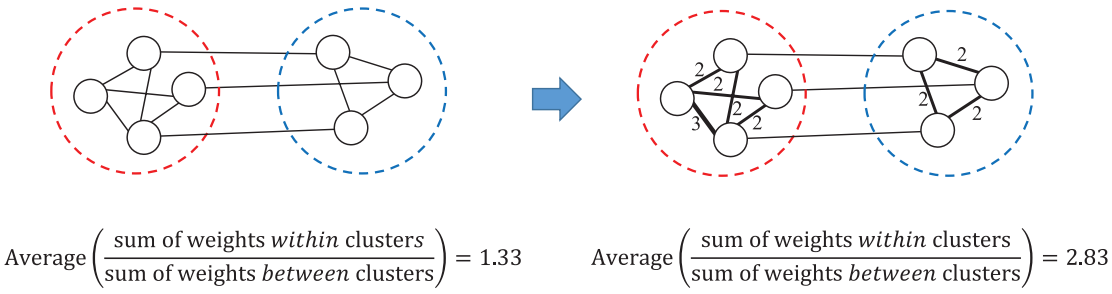


Figure 1. Motivating example for motif-based weighting.

1. We generalize force-directed embedding for graph embedding that allows for modeling higher-order graph substructure.
2. We formally prove the relationship between the proposed method and motif-based cut. Also, we show that our method has an advantage for identifying highly-mixed clusters and is appropriate especially for bipartite graph clustering.
3. We demonstrate, by extensive experiments, that the proposed method outperforms existing methods to find various substructures in graphs.

The rest of the paper is organized as follows. Section 2 reviews the existing embedding methods. Section 3 presents our proposed embedding method based on higher-order graph substructures and formalizes the superiority of the method. Section 4 shows experimental results on both synthetic and real-world networks. Finally, section 5 concludes this study and describes directions for future work.

2. Graph embedding

In this section, we review two widely used *graph embedding* methods: spectral embedding and force-directed embedding. The graph embedding problem has received great attention because it represents the structures of graphs and conveys interesting graph properties using the resulting graph layouts. For a graph data set, graph embedding reveals important relationships in the data set and gives us the big picture without losing the majority of the information itself. Many algorithms for solving the problem have been proposed, and their variants have been applied for various purposes, including knowledge representation, management systems, communication networks, and so on [15].

2.1. Spectral embedding

Spectral clustering is one of the most widely-used clustering algorithms based on embedding techniques [27, 38]. Spectral clustering maps vertices of a given graph to points on a k-dimensional space, and then these points are grouped by a conventional clustering algorithm to form k clusters. Let W be the adjacency matrix (or the affinity matrix) of a graph, D be a n by n diagonal matrix where $d_i = \sum_{j=1}^{n} w_{i,j}$, and n is the number of

vertices in the graph. The embedding step, called *spectral embedding*, finds the top-k eigenvectors of the normalized Laplacian matrix $L = I - D^{-1/2}WD^{-1/2}$ of a given graph according to their eigenvalues. These eigenvectors form the coordinates of vertices on the k-dimensional space. Any conventional clustering algorithm, such as the k-means algorithm, can be performed on the points obtained by embedding to find the communities of the original vertices.

The basis of spectral clustering is as follows. When graph clustering produces a collection of k disjoint cliques, the normalized Laplacian is a block-diagonal matrix that has zero eigenvalues with multiplicity k, and the corresponding eigenvectors are the indicator vectors of membership in the corresponding clusters. Spectral clustering computes k eigenvectors with smallest eigenvalues of the normalized Laplacian and then detects clusters based on the representation of a given graph in the corresponding eigenspace as the output space of embedding. In addition, it is known that this approach is a spectral relaxation of the minimization of the normalized cut [38]. In this respect, spectral embedding is regarded as a typical embedding method for clustering based on graph embedding.

Because of the high complexity of eigenvalue decomposition, it is not easy to apply spectral clustering on large-scale graphs. To resolve this scalability issue, several approximate algorithms have been proposed, and a parallel algorithm with sparsifying the adjacency or similarity matrix of a given graph has recently been proposed [8]. This approach can exhibit nearly linear speedups with the number of processors.

2.2. Force-directed embedding

Force-directed embedding is a popular method for graph drawing, which produces an embedding that typically conforms to some aesthetic criteria. It has received much attention because of its simplicity and intuitiveness for mapping general graphs [39]. Usually, force-directed graph embedding is designed for visualizing the general structure of graphs, but recently it is also used for representing the community structure of graphs [23, 28, 29]. Given an undirected and unweighted graph $\mathcal{G} = (V, E)$, let $p : V \to S$ be a function that maps from a set of vertices to a set of corresponding vertex positions in a low-dimensional space S, e.g. $S = \mathbb{R}^2$ or $S = \mathbb{R}^3$. The force-directed embedding methods regard that two types of forces are exerted on a given graph: (1) *attractive forces* between every two adjacent vertices and (2) *repulsive forces* between every two vertices.

(1) **Attractive force:** For a function p and two vertices i and j, the attractive force exerted on i by j exists. The attractive force is usually modeled by spring forces. It is expressed by $w_{i,j}\|p(i) - p(i)\|^a \overrightarrow{p(i)p(j)}$, where $w_{i,j}$ is a weight of an edge $\{i, j\}$ and $\overrightarrow{p(i)p(j)}$ is a vector pointing from $p(i)$ to $p(j)$. Its magnitude is proportional to the distance between two vertex positions when $a = 1$.

(2) **Repulsive force:** For a function p and two vertices i and j, the repulsive force exerted on i by j exists. The repulsive force is usually modeled by the forces between electrically charged particles. It is expressed by $w_i w_j\|p(i) - p(j)\|^r \overrightarrow{p(j)p(i)}$, where w_i is a weight of a vertex i. Its magnitude is inversely proportional to the distance between positions if $r = -1$.

In this framework, each vertex is regarded as a particle and an edge joining two vertices is modeled by a spring connecting two particles. Then, the force-directed algorithm is designed to find an equilibrium between attractive and repulsive forces. The problem is equivalent to minimizing the energy of the system corresponding to those forces. The energy $E(p|\mathcal{G})$ for the vertex positions $\{p(i)\}_{i\in V}$ for a graph $\mathcal{G}=(V,E)$ is given by equation (1)

$$E(p|\mathcal{G}) = \sum_{\{i,j\}\in E} \frac{w_{i,j}}{a+1} \parallel p(i)-p(j) \parallel^{a+1} - \sum_{\{i,j\}\in V^{(2)}} \frac{w_i w_j}{r+1} \parallel p(i)-p(j) \parallel^{r+1} . \quad (1)$$

Here, the energy is computed by the difference of the sum of attractive energies for every two connected vertices and the sum of repulsive energies for every two vertices, where the derivatives of attractive and repulsive energies correspond to attractive and repulsive forces respectively. The problem is concerned with solving an optimization problem with an objective function in equation (1) over all possible values of p in a given space. Thus, the problem is to locate the positions of vertices that minimize the energy $E(p|\mathcal{G})$.

Among the force-directed embedding algorithms, LinLog ($a=0, r=-1$) [28, 29] is the state-of-the-art algorithm following the above energy minimization problem. When $r=-1$, we take $\ln\|p(i)-p(j)\|$ as the limit of $\|p(i)-p(j)\|^{r+1}/(r+1)$. Although the conventional force-directed embedding algorithms [10, 13] use $w_i \equiv 1$ for all vertices i, LinLog sets w_i to be proportional to the degree of a vertex i, which is called *degree-weighted repulsion*. With degree-weighted repulsion, LinLog produces a layout that is not biased against the high-degree vertices, thereby being appropriate for visualizing complex networks with power-law degree distributions.

The computational complexity of the energy minimization is $\mathcal{O}(|V|^2)$, which is too high for dense graphs. Rather, we can use an approximation algorithm using the Barnes and Hut algorithm that reduces the running time to $\mathcal{O}(|E|+|V|\log|V|)$ [3]. In section 4, we implement force-directed embedding algorithms using the approximation technique.

3. Motif-based embedding

In this section, we propose a new embedding method for graph clustering based on the motifs of graphs. We generalize a classical graph embedding, the force-directed graph embedding, to incorporate the substructures of graphs using motifs. First, we present our proposed graph embedding method, which uses motif-based weighting. Then, we analyze our method by considering the connections between ours and well-known evaluation measures or properties of graph clustering. We also introduce several applications that benefit from wedge-based embedding, which is a special form of motif-based embedding.

3.1. Algorithm

The proposed algorithm for graph embedding is a generalization of force-directed graph embedding using motif-based weighting. First, we introduce the motif-based weighting method for reflecting motif substructures in a given graph. For a (set of) specific type(s) of motifs, we count the numbers of motifs containing two vertices for every

J. Stat. Mech. (2016) 123401

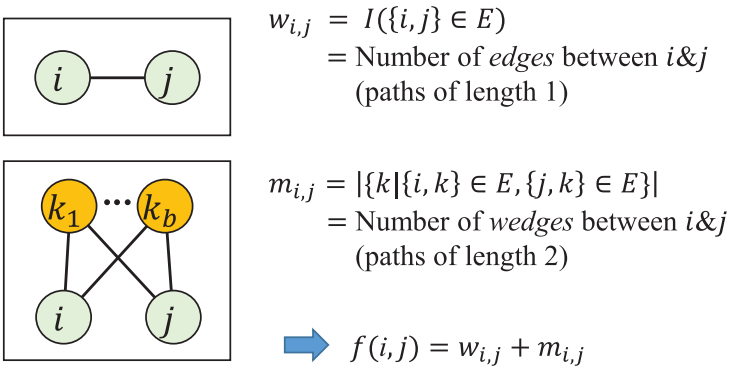


Figure 2. Example of wedge-based weighting.

two vertices. For instance, the third-order substructures in graphs, such as a *triangle* (a clique of size 3) and a *wedge* (a path of length 2), can be regarded as motifs. As an example in figure 2, we regard a wedge between two vertices as a motif, and then count the number of *common neighbors* $m_{i,j}$ between i and j for every pair of vertices. We define the function $f\colon V\times V\to\mathbb{R}$, e.g. $f\colon\{i,j\}\mapsto w_{i,j}+m_{i,j}$, where $w_{i,j}$ is the original weight between i and j, and $m_{i,j}$ is the number of motifs of interest between i and j. If the motif is a wedge, $m_{i,j}$ is formally represented by $m_{i,j}=\sum_k w_{i,k}w_{j,k}$. If the motif is a triangle, because a triangle is a *closed* wedge, $m_{i,j}$ is represented by $m_{i,j}=\sum_k w_{i,j}w_{i,k}w_{j,k}$. One can use nonlinear or another type of function depending on the need of an application in hand. In our experiments, a simple linear combination of edges and one type of motif are used for simplicity.

After the motif-based weighting, we find the layout to minimize the energy using the motif-based weights. In our force-directed embedding, we find the positions $\{p(v)\}_{v\in V}$ that minimize equation (2). There are many embedding methods applicable to minimize the equation [10, 13]

$$E(p|\mathcal{G})=\sum_{\{i,j\}\in E} f(i,j)\,\|p(i)-p(j)\|-\sum_{\{i,j\}\in V^{(2)}} w_i w_j \ln\|p(i)-p(j)\|\,. \tag{2}$$

In order to reduce the complexity, we can use the Barnes–Hut tree approximation to calculate vertex positions of graph embedding [23, 43], where the vertex positions in a region of the tree are processed together to reduce the number of calculations. Then, we apply our motif-based embedding method to graph clustering, in the same fashion as graph clustering with embedding. We can easily find the clusters in the embedding space using the k-means algorithm or other clustering algorithms with the vertex positions and then identify the corresponding clusters of vertices.

3.2. Analysis

We now show the effectiveness of the motif-based embedding for graph clustering in several ways. First, we prove that graph clustering with our motif-based embedding is equivalent to find the optimal clusters based on a measure for evaluating motif-based clusters (section 3.2.1) and is able to find highly-mixed clusters, which are considered to be hard to detect (section 3.2.2). Also, we show that a type of motif-based weighting is a variant of the Jaccard similarity commonly used in graph clustering (section 3.2.3).

3.2.1. Connection with motif-based cut We prove that graph clustering with motif-based embedding solves a relaxation of normalized motif-based cut minimization problem. The motif-based cut, which has been used for graph clustering [17], is a generalization of graph cut that reflects the motifs across clusters. First, we define the motif-based cut and normalized motif-based cut as follows.

Definition 1 (Motif-based cut [17]). Let $\mathcal{G} = (V, E)$ be a connected graph and V_1, V_2 be disjoint subsets of V. Then the motif-based cut between V_1 and V_2 is $\mathrm{mcut}(V_1, V_2) = \sum_{i\in V_1, j\in V_2} f(i, j)$, where $f\colon V\times V\to\mathbb{R}$ is a function defined on a set of pairs of vertices. For instance, $f\colon \{i, j\} \mapsto w_{i,j} + m_{i,j}$, where $w_{i,j}$ is the weight from the original graph and $m_{i,j}$ is the number of motifs that i and j are contained together.

Definition 2 (Normalized motif-based cut). Let $\mathcal{G} = (V, E)$ be a connected graph and V_1, V_2 be disjoint subsets of V. Then the normalized motif-based cut between V_1 and V_2 is $\mathrm{nmcut}(V_1, V_2) = \frac{\mathrm{mcut}(V_1, V_2)}{\deg(V_1)\deg(V_2)}$, where $\deg(V_i)$ is the sum of degrees of vertices belonging to V_i, for $i = 1, 2$.

To simplify theoretical derivation, we assume that the vertices in the same cluster map to the same position and the distance between two positions for V_1 and V_2 in the equilibrium of attractive and repulsive forces is maximized. In theorem 1, we prove that the best cut using our motif-based embedding is obtained when the normalized motif-based cut is minimized.

Theorem 1. *Let $\mathcal{G} = (V, E)$ be a connected graph and (V_1, V_2) be a bipartition of V. Then the distance between two positions is $1/\mathrm{nmcut}(V_1, V_2)$ if there are two possible vertex positions.*

Proof. Let d be the distance between two possible positions obtained by the energy minimization problem with equation (2). Then we have equation (3)

$$\mathcal{E}(p|\mathcal{G}) = d\,\mathrm{mcut}(V_1, V_2) - \deg(V_1)\deg(V_2)\ln d. \tag{3}$$

Differentiating equation (3) with respect to d and setting to zero, we obtain a condition for an equilibrium, $\mathrm{mcut}(V_1, V_2) - \deg(V_1)\deg(V_2)/d = 0$. Therefore, $d = 1/\mathrm{nmcut}(V_1, V_2)$. □

3.2.2. Robustness of motif-based embedding Here, we discuss that our proposed algorithm is more robust to highly-mixed community structure. Clustering with the embedding identifies clusters that are hard to detect. To define the hardness of graph clustering, we introduce the notion of the mixing defined as below.

Definition 3 (Mixing for a vertex). Mixing for a vertex is the ratio between the external degree of the vertex with respect to its cluster and the total degree of the vertex.

Definition 4 (Mixing). Mixing is the fraction of inter-cluster edges defined by the average mixing of vertices.

The hardness of graph clustering is now formalized as follows. Let k_i^{in} and k_i^{out} be the internal and external degree of vertex i with respect to its cluster c and k_c^{in} and k_c^{out} be the sum of internal and external degrees of vertices in c. Also, let $p_{\mathrm{out}} = k_i^{\mathrm{out}}/k_c^{\mathrm{out}}$ and $p_{\mathrm{in}} = k_i^{\mathrm{in}}/k_c^{\mathrm{in}}$. Graph clustering becomes feasible when $p_{\mathrm{in}} > p_{\mathrm{out}}$, which means that the

J. Stat. Mech. (2016) 123401

linking probability inside a community is larger than that between different communities. This condition is a normalized version of the *weak community* [30], which defines a community as a set of vertices such that the sum of their internal degrees exceeds the sum of their external degrees. This condition is equivalent to $\mu < \frac{N-n_c}{N}$ if we assume that μ is the same for all vertices [20], where N is the number of vertices in the graph and n_c is the number of vertices in a cluster c.

In theorem 2, we prove that if we transform a graph to a weighted graph with motif-based weighting then $p_{\text{in}} > p_{\text{out}}$ holds if $n_c < N/2$. The condition $n_c < N/2$ is satisfied in most cases since the size of a cluster is usually smaller than the half of the graph. The theorem shows that the intra-community weights are usually larger than the inter-community weights using motif-based weighting and we conclude that the motif-based weighting method is appropriate for detecting the community structure in graphs.

Theorem 2. *We assume that μ is the same for all vertices and $n_c < N/2$. Then, $p_{\text{in}} > p_{\text{out}}$ for $f(i, j) = m_{i,j}$, where $m_{i,j} = |\{k|\{i,k\} \in E, \{j,k\} \in E\}|$ uses wedge-based weighting.*

Proof. As in [20], we approximate that $k_c^{\text{in}} \sim n_c\langle k\rangle$ and $k_c^{\text{out}} \sim (N-n_c)\langle k\rangle$, where $\langle k\rangle$ is the average degree of the graph. Then, equations (4) and (5) hold

$$p_{\text{in}} \sim \frac{n_c(k_i^{\text{in}}/n_c)^2 + (N-n_c)(k_i^{\text{out}}/(N-n_c))^2}{n_c}, \tag{4}$$

$$p_{\text{out}} \sim \frac{N(k_i^{\text{in}}/n_c)(k_i^{\text{out}}/(N-n_c))}{N-n_c}. \tag{5}$$

From equations (4) and (5), we obtain that $p_{\text{in}} > p_{\text{out}}$ is equivalent to $\frac{N-n_c}{n_c}(k_i^{\text{in}})^2 + (k_i^{\text{out}})^2 > \frac{N}{N-n_c}(k_i^{\text{in}})(k_i^{\text{out}})$. If we set $A = \frac{N-n_c}{n_c}$, $X = k_i^{\text{in}}$, and $Y = k_i^{\text{out}}$, then the last equation is the same with $AX^2 + Y^2 > (1 + 1/A)XY$. By Chebyshev's inequality, the lower bound for the left-hand side is $2\sqrt{A}\,XY$ and $2\sqrt{A}\,XY > (1 + 1/A)XY$ holds for all $A > 1$ since $2\sqrt{A}\,XY$ is increasing and $(1 + 1/A)XY$ is decreasing with respect to A. Therefore, if $\frac{N-n_c}{n_c} > 1$ or, equivalently, $n_c < N/2$, then $p_{\text{in}} > p_{\text{out}}$ holds. □

3.2.3. Connection with Jaccard similarity For graph clustering problems, several weighting methods or similarity measures are used to perform clustering algorithms. Among them, the Jaccard similarity is one of most-widely used similarity metrics to capture information between vertices for the purpose of community detection [36]. We now demonstrate that the Jaccard similarity is related to wedge-based weights $m_{i,j} = |\{k|\{i,k\} \in E, \{j,k\} \in E\}|$. For regular graphs, the orders of the similarity values are the same in the two measures. That is, wedge-based similarity can be regarded as a variant of the Jaccard similarity that does not change the order.

Definition 5 (k-regular graph). A graph is called a k-regular graph if the degrees of vertices are equal to k.

J. Stat. Mech. (2016) 123401

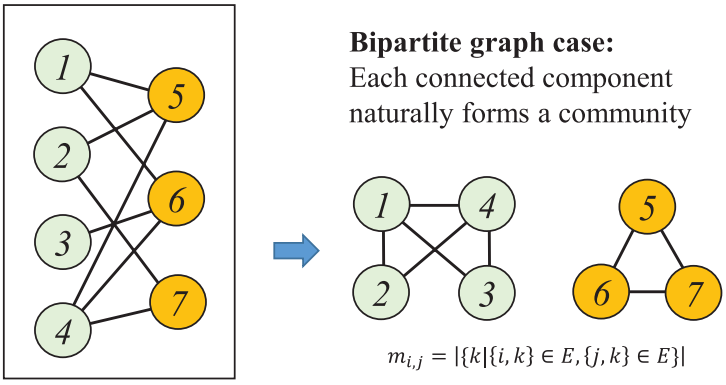


Figure 3. Application for discovering bipartite graph structure.

Theorem 3. *For regular graphs, the wedge-based weights are increasing with respect to the Jaccard similarity.*

Proof. The Jaccard similarity between two vertices i and j is $\sigma(i,j) = \frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|}$, where $\Gamma(i) = \{i\} \cup \{j \in V | \{i,j\} \in E\}$ is the neighborhood of i. For k-regular graphs, $|\Gamma(i) \cup \Gamma(j)| = 2k - |\Gamma(i) \cap \Gamma(j)|$. Also, it is obvious that $m_{i,j} \sim |\Gamma(i) \cap \Gamma(j)|$. Therefore, $\sigma(i,j)$ is increasing with respect to $m_{i,j}$ and vice versa. □

Counting the number of wedges is easier than exactly calculating the Jaccard similarity, but it is still not fast for large graphs. We also discuss a sampling method for approximating the number of wedges and its usefulness for graph clustering in section 3.3.

3.3. Application to detecting bipartite community structure

There have been several community detection methods dedicated to bipartite networks [11, 44]. However, it is hard to support both bipartite and ordinary community structures using a unified framework, because the existence of an edge indicates that the two endpoints do *not* belong to the same cluster in bipartite networks, unlike in ordinary networks. Motif-based embedding can be a solution for this problem, as in figure 3. Using wedges we can induce an embedding for detecting bipartite community structure. In section 4, we empirically verify the superiority of our proposed algorithm for bipartite graphs in both synthetic and real-world graphs.

For a small graph, we are able to easily calculate $m_{i,j} = [A^2]_{i,j}$ for any $i \neq j$. Similarly, we can calculate the values for motifs of the forms of paths or cycles using matrix multiplication. However, in general, matrix multiplication is too slow for dense graphs. The computational complexity for the exact calculation of the number of wedges is $\mathcal{O}(\sum_i k_i^2)$ since there are $\binom{k_i}{2}$ wedges centered at vertex i, where k_i is the degree of i. Instead, we approximate the number of wedges using a wedge sampling method [18]. The sampling method first calculates the degree of each vertex and then chooses a uniform random pair of neighbors of each vertex to generate a wedge. The number of wedges to be sampled from a vertex depends on its degree. Kolda *et al* [18] derived an approximation ratio by which a sparsified graph obtained by the sampling method yields a good

quality for approximating the clustering coefficient, which is closely related to graph clustering. Using this method, we approximate the number of wedges in which two vertices are contained together as a multiple of the number in the sparsified graph by considering the sampling rate.

4. Experiments

We conducted a set of experiments with the proposed graph embedding method on various real-world social networks and synthetic networks. We evaluated the accuracy of the proposed embedding method for graph clustering and discussed its properties with a set of experiments. In total, *six* methods were compared with each other. (1) LinLog is based on force-directed embedding; (2) **LinLog+motif** is based on force-directed embedding with motif-based weighting and is what we propose in this paper; (3) Spectral is based on spectral embedding; (4) Louvain, (5) Infomap, and (6) label propagation are widely recognized as the state-of-the-art methods [20, 23].

(1) LinLog (based on force-directed embedding)

(2) **LinLog+motif** (proposed method)

(3) Spectral clustering (denoted as *Spectral*)

(4) Louvain [6] (based on modularity optimization)

(5) Infomap [33] (based on information theory)

(6) Label propagation [31] (denoted as *LabelProp*)

In order to check the performance of community detection, we applied the k-means clustering algorithm for each result of the embedding techniques (1)–(3).

4.1. Evaluation measures

In order to evaluate the accuracy of community detection algorithms, we used the *Normalized Mutual Information(NMI)* to compare the partition of the set of vertices obtained by the algorithm and the true partition of a given graph [9]. It is a measure of similarity between the partitions, based on information theory. That is, if the ground-truth community structure is known, this measure naturally evaluates the quality of community detection. It ranges from 0 (disagreement) to 1 (agreement) by normalization, and a higher value represents a better quality of community detection. To obtain reliable results, we repeated each test 30 times and reported the average result.

4.2. Network data sets

4.2.1. Real-world networks. The real-world network data sets used for experiments are summarized in table 1. (a) *Football* represents the matches played between U.S. college football teams in the year 2000, where the vertices are grouped to communities according to conferences. (b) *Twitter* is a social network among Irish politicians and political

Table 1. Real-world networks for experiments.

Name	# Nodes	# Links	k	FM_e	FM_w	FM_t	Type
(a) Football	115	613	12	0.643 (394/613)	0.547 (3261/5967)	**0.851** (2067/2430)	Dense triangles
(b) Twitter	343	4831	7	0.634 (3061/4831)	0.459 (115K/251K)	**0.696** (58.6K/84.2K)	Dense triangles
(c) Polblogs	1224	16 715	2	**0.741** (12.4K/16.7K)	0.283 (380K/1.34M)	0.720 (218K/303K)	No special substructure
(d) jdk	6434	127 665	35	**0.308** (39.3K/128K)	0.205 (101M/493M)	0.248 (4.75M/19.1M)	No special substructure
(e) Adjnoun	112	425	2	0.280 (119/425)	**0.396** (2152/5429)	0.169 (144/852)	Near-bipartite
(f) Malaria	1103	2965	2	0 (0/2965)	**1** (25.4K/25.4K)	N/A (none)	Bipartite
(g) Writers	74 775	98 895	2	0 (0/74.8K)	**1** (1.03M/1.03M)	N/A (none)	Bipartite

organizations, where two politician vertices are adjacent if there is a mention between them, and the vertices are assigned to communities according to their political affiliation. (c) *Polblogs* is a network of political blogs around the U.S. election in the year 2004, where two vertices are adjacent if there is a citation between them, and the vertices are grouped by political alignment. (d) *jdk* is a software dependency network, where each vertex represents a class, and an edge between two vertices indicates a dependency between them. (e) *Adjnoun* is a network of common adjective and noun adjacencies for the novel 'David Copperfield', where an edge exists if the words occur in adjacent positions; since most edges occur between adjectives and nouns, the network is approximately bipartite. (f) *Malaria* is a bipartite network, where the vertices correspond to genes and their constituent substrings, and each substring connects to every gene in which it is present. (g) *Writers* is a bipartite authorship network of writers and their articles in DBpedia. We downloaded networks (a), (c) and (e) from the Web, network (b) from [14], network (d) from [41], network (f) from [22], and network (g) from [2]. Note that we used the largest connected components of networks (b) and (g), since they are not connected.

We categorized these networks into three types: networks with dense triangles, (near-) bipartite structure, and no special substructure. Their types were determined by the values of FM_e, FM_w, and FM_t, being respectively the fractions of *edges*, *wedges*, and *triangles* that are connected within the same community. Note that $1 - FM_e$ is based on the concept of mixing, and the other two measures are the generalizations of mixing for motifs. That is, we considered that the characteristic of the community structure of a network is determined by these key local properties. Like many social and information networks which contain many triangles, FM_t is large for (a) Football and (b) Twitter, where a triangle is appropriate for the motif. For (e) near-bipartite and (f)–(g) bipartite networks, FM_w is large, and the a wedge is appropriate for the motif. However, in (c) Polblogs and (d) jdk, FM_e is large, and it seems that there is no special motif substructure; in this case, our motif-based embedding was not shown to be so effective.

The most important task is to determine the appropriate motif substructure *before doing community detection*. For this purpose, we suggest a simple heuristic to determine the motif substructure without knowing the ground-truth community structure. Recall

J. Stat. Mech. (2016) 123401

Table 2. Heuristic for determining the motif substructure.

Name	$P_t^{(\mathcal{G})}$	$P_w^{(\mathcal{G})}$	True Substructure
(a) Football	**0.407**	0.056	Dense triangles
(b) Twitter	**0.335**	0.024	Dense triangles
(c) Polblogs	0.226	0.028	No special substructure
(d) jdk	0.039	0.024	No special substructure
(e) Adjnoun	0.157	**0.111**	Near-bipartite
(f) Malaria	0	**0.186**	Bipartite
(g) Writers	0	**0.378**	Bipartite

Table 3. Accuracy for real-world networks (NMI).

Name	LinLog	LinLog+motif	Spectral	Louvain	Infomap	LabelProp
(a) Football	0.912 ± 0.002	$\mathbf{0.927} \pm 0.002$	0.751 ± 0.044	0.855 ± 0.0000	0.905 ± 0.004	0.895 ± 0.023
(b) Twitter	0.826 ± 0.003	$\mathbf{0.832 \pm 0.002}$	0.538 ± 0.042	0.792 ± 0.0000	0.817 ± 0.0000	0.278 ± 0.264
(c) Polblogs	$\mathbf{0.413 \pm 0.010}$	0.397 ± 0.007	0.393 ± 0.020	0.365 ± 0.0000	0.288 ± 0.004	0.373 ± 0.052
(d) jdk	0.441 ± 0.005	$\mathbf{0.461 \pm 0.007}$	0.172 ± 0.009	0.329 ± 0.0000	0.349 ± 0.004	0.102 ± 0.052
(e) Adjnoun	0.001 ± 0.0000	$\mathbf{0.250} \pm 0.027$	0.004 ± 0.0000	0.002 ± 0.0000	0.003 ± 0.0000	0 ± 0.0000
(f) Malaria	0.529 ± 0.037	$\mathbf{1} \pm 0.0000$	0.006 ± 0.0000	0.007 ± 0.0000	0.008 ± 0.0000	0.008 ± 0.001
(g) Writers	0.018 ± 0.004	$\mathbf{1} \pm 0.0000$	0.007 ± 0.0000	0.015 ± 0.0000	0.017 ± 0.001	0.018 ± 0.005

that calculating FM_e, FM_w, and FM_t in table 1 requires the ground-truth community structure. For a given graph $\mathcal{G}$, let $w(\mathcal{G})$ and $t(\mathcal{G})$ be the number of wedges and triangles, respectively, in $\mathcal{G}$. We use these two measures to calculate the *transitivity* and *intransitivity* of a graph in definitions 6 and 7. Transitivity measures the frequency of triangles (closed wedges), and intransitivity measures that of open wedges.

Definition 6 (Transitivity or global clustering coefficient [26]). For a given graph $\mathcal{G}$, the transitivity of $\mathcal{G}$ is defined by $P_t^{(\mathcal{G})} = t(\mathcal{G})/w(\mathcal{G})$.

Definition 7 (Normalized intransitivity [7]). For a given graph $\mathcal{G}$, the intransitivity of $\mathcal{G}$ is defined by $1 - P_t^{(\mathcal{G})}$. In addition, the normalized intransitivity of $\mathcal{G}$ is $P_w^{(\mathcal{G})} = (1 - P_t^{(\mathcal{G})})/\langle k \rangle$.

Table 2 shows the values of the transitivity ($P_t^{(\mathcal{G})}$) and intransitivity ($P_w^{(\mathcal{G})}$) of the networks. When $P_t^{(\mathcal{G})}$ is relatively high, e.g. >0.3, the appropriate motif is a triangle, as in (a) Football and (b) Twitter. When $P_w^{(\mathcal{G})}$ is relatively high, e.g. >0.1, the appropriate motif is a wedge, as in (e) Adjnoun, (f) Malaria, and (g) Writers. However, if none of the values is high, there is no special substructure, as in (c) Polblogs and (d) jdk.

4.2.2. Synthetic networks. For synthetic networks, we used the LFR benchmark networks [21] which incorporate some important properties found in real-world networks. This allows us to generate a set of graphs with community structure by varying the parameters of the model, including the number of vertices (n), the average degree ($\langle k \rangle$), the maximum degree ($maxK$), the mixing parameters μ (definition 4), the average and maximum sizes of communities, and so on. We generated the networks that have 10 000 and 50 000 vertices and varied the other parameters. Specifically, we set $\langle k \rangle = 20$ and $maxK = 50$ when $n = 10\,000$ and $\langle k \rangle = 100$ and $maxK = 250$ when $n = 50\,000$. The mixing parameter μ is varied from 0.10 to 0.75 by 0.05.

J. Stat. Mech. (2016) 123401

Table 4. *T*-test results for real-world networks.

Name	LinLog	Spectral	Louvain	Infomap	LabelProp
(a) Football	0.0000	0.0000	0.0000	0.0000	0.0000
(b) Twitter	0.0000	0.0000	0.0000	0.0000	0.0000
(c) Polblogs	0.0007	0.3732	0.0000	0.0000	0.0177
(d) jdk	0.0000	0.0000	0.0000	0.0000	0.0000
(e) Adjnoun	0.0000	0.0000	0.0000	0.0000	0.0000
(f) Malaria	0.0000	0.0000	0.0000	0.0000	0.0000
(g) Writers	0.0000	0.0000	0.0000	0.0000	0.0000

Note: All except spectral on polblogs are signicant at the level of 0.05, and most are signicant at the level of machine precision.

Table 5. Gains of each component in motif-based embedding (NMI).

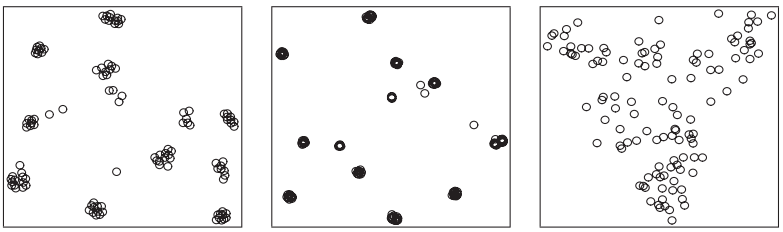
Name	Force-directed embedding	Motif-based weighting	Overall
(a) Football	0.061	0.015	0.076
(b) Twitter	0.220	0.006	0.226
(c) Polblogs	0.058	−0.016	0.042
(d) jdk	0.203	0.020	0.223
(e) Adjnoun	−0.001	0.249	0.248
(f) Malaria	0.522	0.471	0.993
(g) Writers	0.004	0.982	0.986

4.3. Results for real-world networks

Table 3 shows the NMI values for each pair of a network data set and an algorithm. The standard deviation is presented under the NMI value. Overall, LinLog is better than or comparable to Spectral or the state-of-the-art methods (Louvain, Infomap, and LabelProp), and LinLog+motif further increases the accuracy (except for Polblogs, which we will discuss below).

In addition, the statistical significance between the proposed method (LinLog+motif) and each of the others was tested using the two sample Student's *t*-test. The results are summarized in table 4, where the *p*-values are presented. For all networks except Polblogs, the NMI values of the proposed method were significantly different from those of the others. More specifically, the increase from the others including LinLog to LinLog+motif was statistically significant at the significance level of 0.00001. Because Polblogs has no special motif structure, as shown by tables 1 and 2, the difference between LinLog+motif and LinLog was less significant in Polblogs than in the other networks.

In table 5, we separately evaluated the effects of the two main components of the proposed method: (i) force-directed embedding and (ii) motif-based weighting. The former is the difference between the NMI value of LinLog+motif and the average NMI value of the four existing methods (Spectral, Louvain, Infomap, and LabelProp). The latter is the difference between the NMI value of LinLog+motif and that of LinLog. The effects of each component vary depending on the characteristics of the network under consideration. In general, the gains of motif-based weighting become higher in bipartite networks, because a wedge in bipartite networks contains the vertices within the same community whereas an edge does not.



(a) (b) (c)

Figure 4. Embedding results of each algorithm for the football network. (a) LinLog. (b) LinLog+motif. (c) Spectral.

According to the categorization of the networks in table 1, we discuss why motif-based embedding enhances the accuracy of graph clustering or not. The findings from the experiments on real-world networks are summarized as follows.

(i) *Dense triangles within community (Football and Twitter)*: The motif-based weighting is very effective for the case of dense triangles within the same community. Figure 4 visualizes graph embedding of each algorithm. For visualizing the embedding of Spectral in the 2-dimensional space, we used the two principal directions obtained by eigen-decomposition of a normalized Laplacian matrix, while we used the k-dimensional space with the ground-truth $k = 12$ for clustering. Compared to the existing embedding methods (LinLog and Spectral), the vertex positions determined by motif-based embedding (LinLog+motif) are more concentrated around each subregion, which corresponds to each community. When comparing between LinLog+motif and LinLog, since the detectability becomes higher, the NMI values increase by virtue of using the motif-based weighting method. The accuracy of Louvain, Infomap, and LabelProp was shown to be competitive but still lower than that of LinLog+motif.

(ii) *No special substructure (Polblogs and jdk)*: LinLog and LinLog+motif show similar accuracy. The proposed algorithm is more effective if a motif substructure is prevalent in a given graph. In practice, we know that the type of a network we want to analyze and patterns of interest according to the purpose of analysis. It is important to use an appropriate model for analyzing graph structure even if the model is not so complex. If the motif substructure is not prevalent, using motifs does not help improve the quality of community detection as in this case. The accuracy of Louvain, Infomap, and LabelProp was shown to be acceptable but lower than that of LinLog+motif.

(iii) *Bipartite graphs (Adjnoun, Malaria and Writers)*: Using conventional graph clustering algorithms, identifying bipartite graph structure is very hard since the meaning of an edge in bipartite graphs is opposite to that of an edge in general graphs. However, wedge-based embedding enables us to model an existence of strong attraction between vertices in the same community. For (e) Adjnoun, LinLog+motif shows the best performance, but its accuracy is not so high owing to the complex structure of the graph. For (f) Malaria and (g) Writers, the network has a fully bipartite structure, and wedge-based embedding successfully

J. Stat. Mech. (2016) 123401

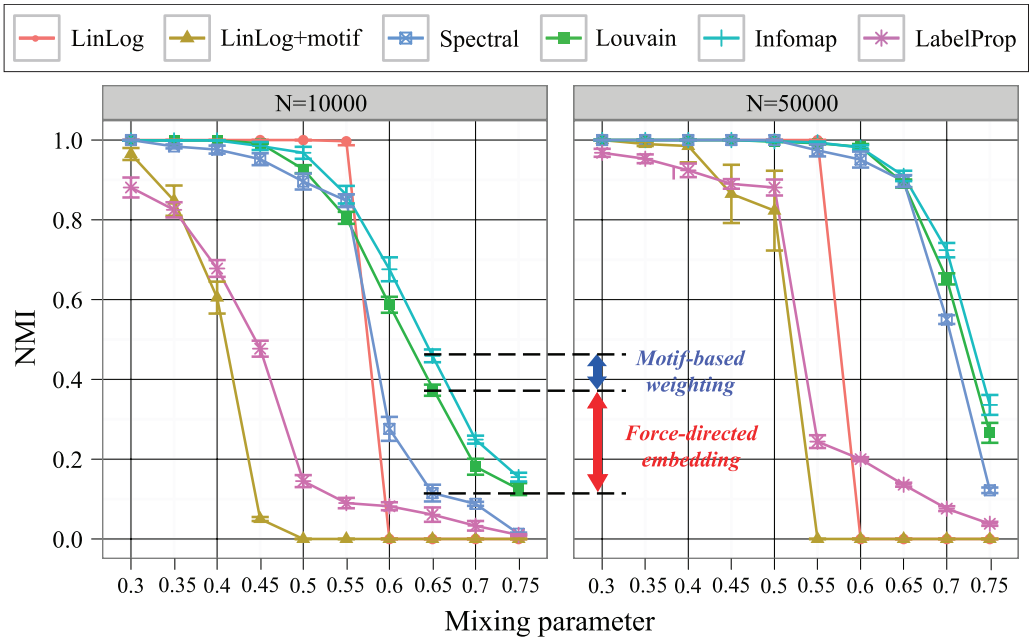


Figure 5. Quality of community detection on synthetic networks.

detected two communities. Without motif-based weighting, the accuracy was low in LinLog, Spectral, Louvain, Infomap, and LabelProp. However, just using the wedge-based weighting, LinLog+motif turned out to be an very effective method.

4.4. Results for synthetic networks

Figure 5 shows the NMI values of the six methods for the LFR networks. The error bars were added using the standard deviation. For the LFR networks, we confirmed that all six methods succeed to detect communities well for small μ values. Previous studies discussed that the transition between undetectable and detectable region begins somewhere between $\mu = 0.4$ and $\mu = 0.5$ [32, 42]. Here, we concentrate on the range of μ from 0.30 to 0.75, which includes a hardly detectable region for community structure.

In figure 5, the accuracy of LinLog+motif is almost always highest among the six methods. As in table 5, the effect of force-directed embedding is measured by the gap between LinLog+motif and the other methods, and that of motif-based weighting is measured by the gap between LinLog+motif and LinLog. The gain by force-directed embedding increased as the mixing (μ) became higher [23], because the accuracy of LinLog degraded less rapidly than those of Spectral, Louvain, Infomap, and LabelProp. We used the motif-based weighting method that the sum of the indicator of an edge and the number of triangles that two end points are contained together. Though the effect of motif-based (i.e. triangles) weighting is not so significant, the motif-based embedding always wins against the original embedding of LinLog. The gap between LinLog + motif and LinLog was kept around 0.1 for high mixing, i.e. when $\mu \geqslant 0.6$. Please note that we did not assume any motif substructure without edges controlled by the parameters of LFR networks. We checked that, as μ increased, the ratio of the average distance between vertices within the same triangle to that between two

adjacent vertices increased more rapidly in the original force-directed embedding than in the motif-based embedding. Specifically, the value increased from 0.13 to 0.77 when μ was varied from 0.30 to 0.75 for LinLog, but the value increased from 0.08 to 0.68 for LinLog+motif. That is, the motif-based embedding locates triangles more closely on the embedding space, and it is useful when there is a tendency that triadic closure occurs within the same community. When such a case happens in networks, our proposed approach will be more powerful.

5. Conclusions

In this paper, we proposed a motif-based embedding method for graph clustering by modeling higher-order relationships among vertices in a graph. Our embedding method considers degree-weighted repulsive forces and motif-based attractive forces to enhance the clustering tendency of points in the output space of graph embedding. The relationship with graph clustering was proven theoretically, and the performance and applicability were verified through extensive experiments. In particular, we checked the superiority of using triangle- and wedge- based embedding for identifying community structure corresponding to such substructures of interest. One can use other motif structures to implement the algorithm, but it is very important to understand and use the proper motif structure in networks.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (2015R1A1A1A05001475).

References

[1] Arenas A, Fernandez A, Fortunato S and Gomez S 2008 Motif-based communities in complex networks *J. Phys. A: Math. Theor.* **41** 224001
[2] Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R and Ives Z 2007 Dbpedia: a nucleus for a web of open data *Proc. of ISWC* pp 722–35
[3] Barnes J and Hut P 1986 A hierarchical *o*(*nlogn*) force-directed calculation algorithm *Nature* **324** 446–9
[4] Benson A, Gleich D and Leskovec J 2015 Tensor spectral clustering for partitioning higher-order network structures *Proc. of SDM* pp 118–26
[5] Benson A, Gleich D and Leskovec J 2016 Higher-order organization of complex networks *Science* **353** 163–6
[6] Blondel V D, Guillaume J-L, Lambiotte R and Lefebvre E 2010 Fast unfolding of communities in large networks *J. Stat. Mech.* P10008
[7] Burt R S 1980 Models of network structure *Annu. Rev. Sociol.* **6** 79–141
[8] Chen W-Y, Song Y, Bai H, Lin C-J and Chang E Y 2010 Parallel spectral clustering in distributed systems *IEEE Trans. Pattern Anal. Mach. Intell.* **33** 568–86
[9] Danon L, Diaz-Guilera A, Duch J and Arenas A 2005 Comparing community structure identification *J. Stat. Mech.* P09008
[10] Davidson R and Harel D 1996 Drawing graphs nicely using simulated annealing *ACM Trans. Graph.* **15** 301–31
[11] Dhillon I S 2003 Co-clustering documents and words using bipartite spectral graph partitioning *Proc. of ACM SIGKDD* pp 269–74

[12] Fortunato S 2010 Community detection in graphs *Phys. Rep.* **486** 75–174
[13] Fruchterman T M J and Reingold E M 1991 Graph drawing by force-directed placement *Softw.: Pract. Exp.* **21** 1129–64
[14] Greene D and Cunningham P 2013 Producing a unified graph representation from multiple social network views *Proc. of WebSci* pp 118–21
[15] Herman I, Melancon G and Marshall M S 2000 Graph visualization and navigation in information visualization: a survey *IEEE Trans. Vis. Comput. Graphics* **6** 24–43
[16] Kim J and Lee J-G 2015 Community detection in multi-layer graphs: a survey *ACM SIGMOD Rec.* **44** 37–48
[17] Klymko C, Gleich D F and Kolda T G 2014 Using triangles to improve community detection in directed networks *Proc. of ASE BigDataScience*
[18] Kolda T G, Pinar A and Seshadhri C 2013 Triadic measures on graphs: the power of wedge sampling *Proc. of SDM* pp 10–18
[19] Kossinets G and Watts D J 2006 Empirical analysis of an evolving social network *Science* **311** 88–90
[20] Lancichinetti A and Fortunato S 2009 Community detection algorithms: a comparative analysis *Phys. Rev.* E **80** 056117
[21] Lancichinetti A, Fortunato S and Radicchi F 2008 Benchmark graphs for testing community detection algorithms *Phys. Rev.* E **78** 046110
[22] Larremore D B, Clauset A and Buckee C O 2013 A network approach to analyzing highly recombinant malaria parasite genes *PLoS Comput. Biol.* **9** e1003268
[23] Lim S, Kim J and Lee J-G 2016 BlackHole: robust community detection inspired by graph drawing *Proc. of IEEE ICDE* pp 25–36
[24] Lim S, Ryu S, Kwon S, Jung K and Lee J-G 2014 LinkSCAN: overlapping community detection using the link-space transformation *Proc. of IEEE ICDE* pp 292–303
[25] Newman M E J 2013 Spectral methods for community detection and graph partitioning *Phys. Rev.* E **88** 042822
[26] Newman M E J, Watts D J and Strogatz S H 2002 Random graph models of social networks *Proc. Natl Acad. Sci. USA* **99** 2566–72
[27] Ng A Y, Jordan M I and Weiss Y 2001 On spectral clustering: analysis and an algorithm *Proc. of NIPS* pp 849–56
[28] Noack A 2007 Energy models for graph clustering *J. Graph. Algebr. Appl.* **11** 453–80
[29] Noack A 2009 Modularity clustering is force-directed layout *Phys. Rev.* E **79** 026102
[30] Radicchi F, Castellano C, Cecconi F, Loreto V and Parisi D 2004 Defining and identifying communities in networks *Proc. Natl Acad. Sci. USA* **101** 2658–63
[31] Raghavan U N, Albert R and Kumara S 2007 Near linear time algorithm to detect community structures in large-scale networks *Phys. Rev.* E **76** 036106
[32] Reichardt J and Leone M 2008 (Un)detectable cluster structure in sparse networks *Phys. Rev. Lett.* **101** 078701
[33] Rosvall M and Bergstrom C T 2008 Maps of random walks on complex networks reveal community structure *Proc. Natl Acad. Sci. USA* **105** 1118–23
[34] Rosvall M and Bergstrom C T 2011 Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems *PLoS One* **6** e18209
[35] Rosvall M, Esquivel A V, Lancichinetti A, West J D and Lambiotte R 2014 Memory in network flows and its effects on spreading dynamics and community detection *Nat. Commun.* **5** 4630
[36] Ruan Y, Fuhry D and Parthasarathy S 2013 Efficient community detection in large networks using content and links *Proc. of WWW* pp 1089–98
[37] Schaeffer S E 2007 Graph clustering *Comput. Sci. Rev.* **1** 27–64
[38] Shi J and Malik J 2000 Normalized cuts and image segmentation *IEEE Trans. Pattern Anal. Mach. Intell.* **22** 888–905
[39] Tamassia R 2013 *Handbook of Graph Drawing and Visualization* (London: Chapman and Hall/CRC)
[40] von Luxburg U 2007 A tutorial on spectral clustering *Stat. Comput.* **17** 395–416
[41] Šubelj L and Bajec M 2011 Community structure of complex software systems: analysis and applications *Physica* A **390** 2968–75
[42] Wang C, Wang H, Liu J, Ji M, Su L, Chen Y and Han J 2013 On the detectability of node grouping in networks *Proc. of SDM* pp 713–21
[43] Yang Z, Peltonen J and Kaski S 2014 Optimization equivalence of divergences improves neighbor embedding *Proc. of ICML* pp 460–8
[44] Zha H, He X, Ding C, Simon H and Gu M 2001 Bipartite graph partitioning and data clustering *Proc. of ACM CIKM* pp 25–32