

A Novel Weighted Distance Measure for Multi-Attributed Graph

Muhammad Abulaish, *SMIEEE*

Department of Computer Science
South Asian University
New Delhi-21, India
abulaish@sau.ac.in

Jahiruddin*

Department of Computer Science
Jamia Millia Islamia
New Delhi-25, India
jahir.jmi@gmail.com

ABSTRACT

Due to exponential growth of complex data, graph structure has become increasingly important to model various entities and their interactions, with many interesting applications including, bioinformatics, social network analysis, etc. Depending on the complexity of the data, the underlying graph model can be a simple directed/undirected and/or weighted/un-weighted graph to a complex graph (aka multi-attributed graph) where vertices and edges are labelled with multi-dimensional vectors. In this paper, we present a novel weighted distance measure based on weighted Euclidean norm which is defined as a function of both vertex and edge attributes, and it can be used for various graph analysis tasks including classification and cluster analysis. The proposed distance measure has flexibility to increase/decrease the weightage of edge labels while calculating the distance between vertex-pairs. We have also proposed a *MAGDist* algorithm, which reads multi-attributed graph stored in CSV files containing the list of vertex vectors and edge vectors, and calculates the distance between each vertex-pair using the proposed weighted distance measure. Finally, we have proposed a multi-attributed similarity graph generation algorithm, *MAGSim*, which reads the output of *MAGDist* algorithm and generates a similarity graph that can be analysed using classification and clustering algorithms. The significance and accuracy of the proposed distance measure and algorithms is evaluated on Iris and Twitter data sets, and it is found that the similarity graph generated by our proposed method yields better clustering results than the existing similarity graph generation methods.

CCS CONCEPTS

• **Information systems** → **Similarity measures**; *Data analytics*; *Clustering*; • **Human-centered computing** → Social network analysis;

KEYWORDS

Data mining, Clustering, Multi-attributed graph, Weighted distance measure, Similarity measure

* To whom correspondence should be made

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Compute '17, Bhopal, India

© 2017 ACM. 978-1-4503-5323-6/17/11...\$15.00

DOI: 10.1145/3140107.3140114

ACM Reference format:

Muhammad Abulaish, *SMIEEE* and Jahiruddin. 2017. A Novel Weighted Distance Measure for Multi-Attributed Graph. In *Proceedings of 10th Annual ACM India Compute Conference, Bhopal, India, November 16–18, 2017 (Compute '17)*, 9 pages.
DOI: 10.1145/3140107.3140114

1 INTRODUCTION

Due to increasing popularity of Internet and Web2.0, the User-Generated Contents (UGC)s are growing exponentially. Since most of the UGCs are not independent, rather linked, the graph data structure is being considered as a suitable mathematical tool to model the inherent relationships among data. Simple linked data are generally modelled using simple graph $G = (V, E)$, where V is the set of vertices representing key concepts or entities and $E \subseteq V \times V$ is the set of links between the vertices representing the relationships between the concepts or entities. Depending on the nature of data to be modelled, the graph G could be weighted/un-weighted or directed/undirected, and it may have self-loops. However, there are many complex data like online social networks, where an entity is characterized by a set of features and multiple relationships exist between an entity-pair. To model such data, the concept of multi-attributed graph can be used wherein each vertex is represented by an n -dimensional vector and there may be multiple weighted edges between each pair of vertices. In other words, in a multi-attributed graph, both vertices and edges are assigned multiple labels. One of the important tasks related to graph data analysis is to decompose a given graph into multiple cohesive sub-graphs, called clusters, based on some common properties. The clustering is an unsupervised learning process to identify the underlying structure (in the form of clusters) of data, which is generally based on some similarity/distance measures between data elements. Graph clustering is special case of clustering process which divides an input graph into a number of connected components (sub-graphs) such that intra-component edges are maximum and inter-components edges are minimum. Each connected component is called a cluster (aka community) [13]. Though graph clustering has received attention of many researchers and a number of methods for graph clustering has been proposed by various researchers [19], to the best of our knowledge, the field of clustering multi-attributed graph is still unexplored.

Since similarity/distance measure is the key requirement for any clustering algorithm, in this paper, we have proposed a novel weighted distance measure based on weighted Euclidean norm to calculate the distance between the vertices of a multi-attributed graph. The proposed distance measure considers both vertex and edge weight-vectors, and it is flexible enough to assign different weightage to different edges and scale the overall edge-weight

while computing the weighted distance between a vertex-pair. We have also proposed a *MAGDist* algorithm that reads the lists of vertex and edge vectors as two separate CSV files and calculates the distance between each vertex-pairs using the proposed weighted distance measure. Finally, we have proposed a multi-attributed similarity graph generation algorithm, *MAGSim*, which reads the output of *MAGDist* algorithm and generates a similarity graph. In other words, *MAGDist* and *MAGSim* algorithms can be used to transform a multi-attributed graph into a simple weighted similarity graph, wherein a single weighted edge exists between each vertex-pair. Thereafter, the weighted similarity graph can be analysed using existing classification and clustering algorithms for varied purposes. The efficacy of the proposed distance measure and algorithms is tested over the well-known Iris data set and a Twitter data set related to three different events. The proposed similarity graph generation approach is compared with other existing similarity graph generation methods like Gaussian kernel and k -nearest neighbours methods, and it is found that our proposed approach yields better clustering results in comparison to the existing methods. Moreover, the proposed distance measure can be applied to any graph data where both vertices as edges are multi-dimensional real vectors. In case, the data is not linked, i.e., edges do not exist between the edges, the proposed distance measure simple works as an Euclidean distance between the node vectors.

The rest of the paper is organized as follows. Section 2 presents a brief review on various distance measures and graph clustering methods. Section 3 presents the formulation of our proposed weighted distance measure for multi-attributed graph. It also presents the founding mathematical concepts and formal descriptions of the *MAGDist* and *MAGSim* algorithms. Section 4 presents the experimental setup and evaluation results. Finally, section 5 concludes the paper with future directions of work.

2 RELATED WORK

Multi-attributed graph is used to model many complex problems, mainly those in which objects or entities are characterized using a set of features and linked together in different ways. For example, an online social network user can be characterized using a set of features like ID, display name, demographic information, interests, etc. Similarly, two person in an online social network may be associated through different relationships like friendship, kinship, common interests, common friends, etc [4]. In [15], the authors proposed a new principled framework for estimating graphs from multi-attribute data. Their method estimates the partial canonical correlations that naturally accommodate complex vertex features instead of estimating the partial correlations. However, when the vertices of a multi-attributed graph have different dimensions, it is unclear how to combine the estimated graphs to obtain a single Markov graph reflecting the structure of the underlying complex multi-attributed data. In [14], Katenka and Kolaczyk proposed a method for estimating association networks from multi-attribute data using canonical correlation as a dependence measure between two groups of attributes.

Clustering is an unsupervised learning technique which aims to partition a data set into smaller groups in which elements of a group are similar and that elements from different groups are dissimilar.

As a result, clustering has broad applications, including the analysis of business data, spatial data, biological data, social network data, and time series data [24], and a large number of researchers have targeted clustering problem [2, 12, 17]. Since graph is a popular data structure to model structural as well as contextual relationships of data objects, recently graph data clustering is considered as one of the interesting and challenging research problems [16, 22], and it aims to decompose a large graph into different densely connected components. Some of the applications of the graph clustering is community detection in online social networks [5, 7, 8], social bot detection [10, 11], spammer detection [1, 3, 6, 9], functional relation identification in large protein-protein interaction networks, and so on. Generally graph clustering methods are based on the concept of normalized cut, structural density, or modularity[16, 22]. However, like [20], graphs can be partitioned on the basis of attribute similarity in such a way that vertices having similar attribute vectors are grouped together to form a cluster. Mainly graph clustering and simple data clustering differs in the way associations between objects are calculated. In graph clustering, the degree of association between a pair of objects is calculated as closeness between the respective nodes of the graph, generally in terms of number of direct links or paths between them. Whereas, in simple data clustering, the degree of association between a pair of objects is calculated in terms of similarity/distance measure between the vector representations of the objects. Both topological structure and vertex properties of a graph play an important role in many real applications. For example, in a social graph, vertex properties can be used to characterize network users, whereas edges can be used to model different types of relationships between the users.

It can be seen from the discussions mentioned above that most of the graph analysis approaches have considered only one aspect of the graph structure and ignored the other aspects, due to which the generated clusters either have loose intra-cluster structures or reflect a random vertex properties distribution within clusters. However, as stated in [24], "an ideal graph clustering should generate clusters which have a cohesive intra-cluster structure with homogeneous vertex properties, by balancing the structural and attribute similarities". Therefore, considering both vertex attributes and links for calculating similarity/distance between the pairs of vertices in a graph is one of basic requirements for clustering complex multi-attributed graphs. In this paper, we have proposed a weighted distance function that can transform a multi-attributed graph into a simple similarity graph on which existing graph clustering algorithms can be applied to identify densely connected components.

3 PRELIMINARIES AND DISTANCE MEASURES

In this section, we present the mathematical basis and formulation of the proposed weighted distance measures for multi-attributed graph. Starting with the basic mathematical concepts in subsection 3.1, the formulation of the proposed distance function along with an example is presented in the subsequent subsections.

3.1 Founding Mathematical Concepts

Since the Linear Algebra concepts *inner product* and *norm* generally form the basis for any distance function, we present a brief overview of these mathematical concepts in this section.

Inner Product: An *inner product* is a generalized form of the *vector dot product* to multiply vectors together in such a way that the end result is a scalar quantity. If $\vec{a} = (a_1, a_2, \dots, a_n)^T$ and $\vec{b} = (b_1, b_2, \dots, b_n)^T$ are two vectors in the vector space \mathfrak{R}^n , then the *inner product* of \vec{a} and \vec{b} is denoted by $\langle \vec{a}, \vec{b} \rangle$ and defined using equation 1 [18].

$$\langle \vec{a}, \vec{b} \rangle = \vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + \dots + a_n b_n = \sum_{i=1}^n a_i b_i \quad (1)$$

$$\langle \vec{a}, \vec{b} \rangle = \vec{a}^T \vec{b} = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (2)$$

$$= [a_1 b_1 + a_2 b_2 + \dots + a_n b_n]$$

Alternatively, *inner product* of \vec{a} and \vec{b} can be defined as a matrix product of row vector \vec{a}^T and column vector \vec{b} using equation 2. However, as stated in [18], an inner product must satisfy the following four basic properties, where \vec{a} , \vec{b} , and \vec{c} belong to \mathfrak{R}^n , and n is a *scalar* quantity.

- (1) $\langle \vec{a} + \vec{b}, \vec{c} \rangle = \langle \vec{a}, \vec{c} \rangle + \langle \vec{b}, \vec{c} \rangle$
- (2) $\langle n\vec{a}, \vec{b} \rangle = n\langle \vec{a}, \vec{b} \rangle$
- (3) $\langle \vec{a}, \vec{b} \rangle = \langle \vec{b}, \vec{a} \rangle$
- (4) $\langle \vec{a}, \vec{a} \rangle > 0$ and $\langle \vec{a}, \vec{a} \rangle = 0$ iff $\vec{a} = 0$

Weighted Inner Product: The *weighted inner product* is generally used to emphasize certain features (dimensions) through assigning weight to each dimension of the vector space \mathfrak{R}^n . If $d_1, d_2, \dots, d_n \in \mathfrak{R}$ are n positive real numbers between 0 and 1 and D is a corresponding diagonal matrix of order $n \times n$, then the *weighted inner product* of vectors \vec{a} and \vec{b} is defined by equation 3 [18]. It can be easily verified that *weighted inner product* also satisfies the four basic properties of the *inner product* mentioned earlier in this section.

$$\langle \vec{a}, \vec{b} \rangle_D = \vec{a}^T D \vec{b} = \sum_{i=1}^n d_i a_i b_i, \text{ where } D = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_n \end{bmatrix} \quad (3)$$

Norm: In linear algebra and related area of mathematics, the *norm* on a vector space \mathfrak{R}^n is a function used to assign a non negative real number to each vector $\vec{a} \in \mathfrak{R}^n$. Every inner product gives a *norm* that can be used to calculate the length of a vector. However, not every *norm* is derived from an inner product [18]. The *norm* of a vector $\vec{a} \in \mathfrak{R}^n$ is denoted by $\|\vec{a}\|$ and can be defined using equation 4.

$$\|\vec{a}\| = \sqrt{\langle \vec{a}, \vec{a} \rangle} \quad (4)$$

As given in [18], if $\vec{a} = (a_1, a_2, \dots, a_n)^T$ and $\vec{b} = (b_1, b_2, \dots, b_n)^T$ are two vectors of the vector space \mathfrak{R}^n and $n > 0$ is a scalar quantity, then every *norm* satisfies the following three properties.

- (1) $\|\vec{a}\| > 0$ with $\|\vec{a}\| = 0$ iff $\vec{a} = 0$ (*positivity*)
- (2) $\|n\vec{a}\| = n\|\vec{a}\|$ (*homogeneity*)
- (3) $\|\vec{a} + \vec{b}\| \leq \|\vec{a}\| + \|\vec{b}\|$ (*triangle inequality*)

Some of the well-known norms defined in [21] are as follows:

- (1) *L₁-norm (aka Manhattan norm):* The *L₁-norm* of a vector $\vec{a} \in \mathfrak{R}^n$ is simply obtained by adding the absolute values of its components. Formally, the *L₁-norm* of the vector \vec{a} is represented as $\|\vec{a}\|_1$ and it can be defined using equation 5.

$$\|\vec{a}\|_1 = \sum_{i=1}^n |a_i| \quad (5)$$

- (2) *L₂-norm (aka Euclidean norm):* The *L₂-norm* of a vector $\vec{a} \in \mathfrak{R}^n$ is obtained by taking the positive square root of the sum of the square of its components. Formally, the *L₂-norm* of a vector \vec{a} is represented as $\|\vec{a}\|_2$ and it can be defined using equation 6.

$$\|\vec{a}\|_2 = \left(\sum_{i=1}^n a_i^2 \right)^{1/2} \quad (6)$$

- (3) *Infinity-norm (aka max-norm):* The *Infinity-norm* of a vector $\vec{a} \in \mathfrak{R}^n$ is obtained as maximum of the absolute values of the components of the vector. Formally, the *infinity-norm* of the vector \vec{a} is represented as $\|\vec{a}\|_\infty$ and it can be defined using equation 7.

$$\|\vec{a}\|_\infty = \max_{i=1}^n |a_i| \quad (7)$$

- (4) *L_p-norm:* The *L_p norm* of a vector $\vec{a} \in \mathfrak{R}^n$ is the positive p^{th} root of the sum of p^{th} power of the absolute value of the components of the vector. Formally, the *L_p-norm* of the vector \vec{a} is represented as $\|\vec{a}\|_p$ and it can be defined using equation 8.

$$\|\vec{a}\|_p = \left(\sum_{i=1}^n |a_i|^p \right)^{1/p} \quad (8)$$

- (5) *Weighted Euclidean norm:* The *weighted Euclidean norm* of a vector $\vec{a} \in \mathfrak{R}^n$ is a special case of the Euclidean norm in which different dimensions of \vec{a} can have different weights. If $A^T = \vec{a}^T$ is a row matrix of order $1 \times n$, and D is a diagonal matrix of order $n \times n$ whose i^{th} diagonal element is the weight of the i^{th} dimension of the vector \vec{a} , then the weighted Euclidean norm of \vec{a} is the positive square root of the product matrix $A^T D A$. In case D is an identity matrix, this gives simply the *Euclidean norm*. Formally, for a given vector $\vec{a} \in \mathfrak{R}^n$ and a weight (diagonal) matrix D of order $n \times n$, the *weighted Euclidean norm* of \vec{a} is represented as $\|\vec{a}\|_D$ and defined using equation 9.

$$\|\vec{a}\|_D = \left(A^T D A \right)^{1/2} = \left(\sum_{i=1}^n d_i a_i^2 \right)^{1/2} \quad (9)$$

3.2 Multi-Attributed Graph

In this section, we present a formal definition of multi-attributed graph, in which both vertices and edges can be represented as multi-dimensional vectors. Mathematically, a multi-attributed graph can

be defined as a quadruple $G_m = (V, E, \mathcal{L}_v, \mathcal{L}_e)$, where $V \neq \phi$ represents the set of vertices, $E \subseteq V \times V$ represents the set of edges, $\mathcal{L}_v : V \rightarrow \mathfrak{R}^n$ is a vertex-label function that maps each vertex to an n -dimensional real vector, and $\mathcal{L}_e : E \rightarrow \mathfrak{R}^m$ is an edge-label function that maps each edge to an m -dimensional real vector. Accordingly, a vertex $v \in V$ in a multi-attributed graph can be represented as an n -dimensional vector $\vec{v} = (v_1, v_2, \dots, v_n)^T$ and an edge between a vertex-pair (u, v) can be represented as an m -dimensional vector $\vec{e}(u, v) = (e_1(u, v), e_2(u, v), \dots, e_m(u, v))^T$.

3.3 Proposed Weighted Distance Measure

In this section, we present our proposed weighted distance measure for multi-attributed graph that is based on *weighted Euclidean norm*. If $u = \vec{u} = (u_1, u_2, \dots, u_n)^T$ and $v = \vec{v} = (v_1, v_2, \dots, v_n)^T$ are two vertices of a multi-attributed graph and $(u, v) = \vec{e}(u, v) = (e_1(u, v), e_2(u, v), \dots, e_m(u, v))^T$ is a multi-labelled edge between u and v , then the distance between u and v is denoted by $\Delta(u, v)$ and calculated using equation 10, where λ is a scalar value (equation 11), and I is an identity matrix of order $n \times n$. The value of λ depends on the aggregate weight, $\omega(u, v)$, of the edges between the vertex-pair (u, v) , which is calculated using equation 12. The value of $\gamma > 0$ is a user-supplied threshold, providing flexibility to tune the value of λ for calculating distance between a vertex-pair. In equation 12, α_i is a constant such that $\alpha_i \geq 0$ and $\sum_{i=1}^n \alpha_i = 1$.

$$\Delta(u, v) = \Delta(\vec{u}, \vec{v}) = \|\vec{u} - \vec{v}\|_{\lambda I} = \left((u - v)^T \cdot \lambda I \cdot (u - v) \right)^{1/2}$$

$$= \left(\begin{bmatrix} u_1 - v_1 & \dots & u_n - v_n \end{bmatrix} \begin{bmatrix} \lambda & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda \end{bmatrix} \begin{bmatrix} u_1 - v_1 \\ \vdots \\ u_n - v_n \end{bmatrix} \right)^{1/2} \quad (10)$$

$$= \left(\lambda(u_1 - v_1)^2 + \lambda(u_2 - v_2)^2 + \dots + \lambda(u_n - v_n)^2 \right)^{1/2}$$

$$= \sqrt{\lambda} \times \left(\sum_{i=1}^n (u_i - v_i)^2 \right)^{1/2}$$

$$\lambda = \frac{1}{(1 + \omega(u, v))^\gamma} \quad (11)$$

$$\omega(u, v) = \alpha_1 e_1(u, v) + \alpha_2 e_2(u, v) + \dots + \alpha_m e_m(u, v)$$

$$= \sum_{i=1}^m \alpha_i e_i(u, v) \quad (12)$$

It may be noted that the λ function defined using equation 11 is a monotonic decreasing function, as proved in theorem 3.1, so that the distance between a vertex-pair could decrease with increasing ties between them, and vice-versa. The novelty of the proposed distance function lies in providing flexibility (i) to assign different weights to individual edges using α , and (ii) to control the degree of monotonicity using γ , as shown in figure 1. It may also be noted that the value of λ will always be 1, if either if $\omega(u, v) = 0$ or $\gamma = 0$.

THEOREM 3.1. *The λ function defined in equation 11 is a monotonically decreasing function.*

PROOF. Let $\lambda = f(\omega)$ be a function of ω , where ω is the aggregate weight of edges between a vertex-pair (u, v) .

Let $\omega_1, \omega_2 \geq 0$ such that $\omega_1 \leq \omega_2$

$$\Rightarrow (1 + \omega_1) \leq (1 + \omega_2)$$

$$\Rightarrow (1 + \omega_1)^\gamma \leq (1 + \omega_2)^\gamma, \text{ where } \gamma \geq 1$$

$$\Rightarrow \frac{1}{(1 + \omega_1)^\gamma} \geq \frac{1}{(1 + \omega_2)^\gamma}$$

$$\Rightarrow f(\omega_1) \geq f(\omega_2)$$

Hence, λ is a monotonically decreasing function of ω for $\gamma \geq 1$.

Similarly, it can be shown that λ is a monotonically decreasing function of γ for $\omega > 0$.

Let $\lambda = f(\gamma)$ be a function of γ

Let $\gamma_1, \gamma_2 \geq 1$ such that $\gamma_1 \leq \gamma_2$

$$\Rightarrow (1 + \omega)^{\gamma_1} \leq (1 + \omega)^{\gamma_2}, \text{ where } \omega > 0$$

$$\Rightarrow \frac{1}{(1 + \omega)^{\gamma_1}} \geq \frac{1}{(1 + \omega)^{\gamma_2}}$$

$$\Rightarrow f(\gamma_1) \geq f(\gamma_2)$$

Hence, λ is a monotonically decreasing function of γ for $\omega > 0$. \square

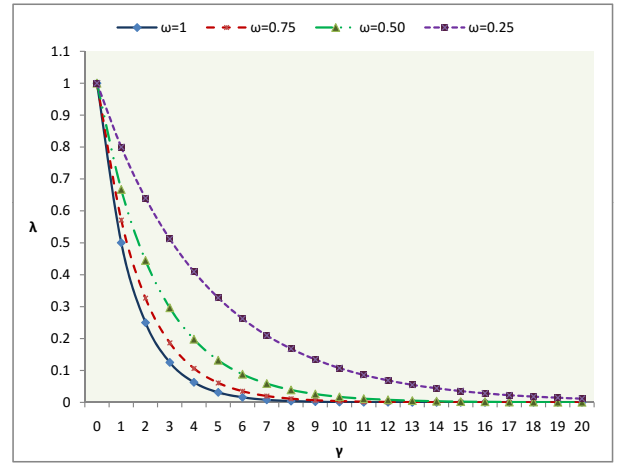


Figure 1: Visualization of the monotonically decreasing nature of λ for varying γ and ω values

It should be noted that if there is no direct link (edge) between a vertex pair (u, v) , then the value of $\omega(u, v)$ in equation 11 becomes zero, leading the value of λ to 1. In this case, the value of $\Delta(u, v)$ is simply an *Euclidean distance* between the vertex-pair (u, v) , as proved in theorem 3.2.

THEOREM 3.2. *In a multi-attributed graph, if there is no edge between a vertex-pair then the distance between them is simply an Euclidean distance.*

PROOF. Let $u = \vec{u} = (u_1, u_2, \dots, u_n)^T$ and $v = \vec{v} = (v_1, v_2, \dots, v_n)^T$ be two vertices not connected by any edge. Since there is no edge between the vertex-pair (u, v) , the edge vector $\vec{e}(u, v) = (e_1(u, v), e_2(u, v), \dots, e_m(u, v))^T = \vec{0}$.

$$\Rightarrow e_1(u, v) = e_2(u, v) = \dots = e_m(u, v) = 0$$

$$\text{Hence, } \omega(u, v) = \alpha_1 \cdot e_1(u, v) + \alpha_2 \cdot e_2(u, v) + \dots + \alpha_m \cdot e_m(u, v)$$

$$= \alpha_1 \cdot 0 + \alpha_2 \cdot 0 + \dots + \alpha_m \cdot 0 = 0. \quad [\text{using equation 12}]$$

$$\text{Hence, } \lambda = \frac{1}{(1 + \omega(u, v))^\gamma} = \frac{1}{(1 + 0)^\gamma} = 1 \quad [\text{using equation 11}]$$

$$\text{Finally, } \Delta(u, v) = \sqrt{\lambda} \times \left(\sum_{i=1}^n (u_i - v_i)^2 \right)^{1/2} \quad [\text{using equation 10}]$$

$$= \sqrt{1} \times \left(\sum_{i=1}^n (u_i - v_i)^2 \right)^{1/2} = \left(\sum_{i=1}^n (u_i - v_i)^2 \right)^{1/2}, \text{ which is an Euclidean distance between the vertex-pair } (u, v). \quad \square$$

Algorithm 1 presents a formal way to calculate the distance between all pairs of vertices of a given multi-attributed graph using *MAGDist*. The proposed *MAGDist* algorithm reads a multi-attributed graph using two separate CSV files – one containing the list of vertex vectors and the other containing the list of edge vectors, and produces the distance between each vertex-pairs as a CSV file, wherein each tuple contains a vertex-pair and distance value. We have also proposed *MAGSim* algorithm to generate similarity graph using the distance values calculated by the *MAGDist* algorithm. The proposed algorithm reads each vertex-pair and its distance value $\langle i, j, \Delta(i, j) \rangle$ and calculates similarity between the vertex-pair (i, j) using equation 13.

$$\text{sim}(i, j) = 1 - \frac{\Delta(i, j)}{\max_{x, y \in V} \{\Delta(x, y)\}} \quad (13)$$

Algorithm 1: MAGDist $(\mathcal{L}_v, \mathcal{L}_e, \alpha, \gamma)$

```

// computing distance between all pairs of
vertices of a multi-attributed graph
Input : CSV files  $\mathcal{L}_v$  and  $\mathcal{L}_e$  containing list of vertex
vectors, and edge vectors, respectively. An 1D
array  $\alpha[1..m]$ , wherein  $\alpha_i \geq 0$  and  $\sum_{i=1}^m \alpha_i = 1$  for
calculating the linear combination of edge weights
between a pair of vertices. A positive integer
threshold  $\gamma$  representing the weightage of edges in
distance calculation.
Output: A CSV file  $D$  containing distance between each
pairs of vertices.
1  $n_v \leftarrow \text{vertexCount}[\mathcal{L}_v]$ ; // number of vertices
2  $n \leftarrow \text{vertexDimCount}[\mathcal{L}_v]$ ; // vertex vector
dimension
3  $m \leftarrow \text{edgeDimCount}[\mathcal{L}_e]$ ; // edge vector dimension
4  $V[n_v][n] \leftarrow \mathcal{L}_v$ ; // reading  $\mathcal{L}_v$  into array  $V$ .
5 for each vertex-pair  $(i, j) \in \mathcal{L}_e$  do
    // calculating aggregate weight  $\omega(i, j)$  of the
    edges between vertex-pair  $(i, j)$ .
6  $\omega(i, j) \leftarrow 0$ ;
7 for  $k \leftarrow 1$  to  $m$  do
8 |  $\omega(i, j) = \omega(i, j) + \alpha[k] \times e_k(i, j)$ ; // [eqn. 12]
9 end
    // calculating the value of scalar quantity  $\lambda$ 
10  $\lambda = \frac{1}{(1+\omega(i, j))^\gamma}$ ; // [eqn. 11]
    // calculating distance  $\Delta(i, j)$  between the
    vertex-pair  $(i, j)$ .
11  $d \leftarrow 0$ ;
12 for  $k \leftarrow 1$  to  $n$  do
13 |  $d = d + (V[i][k] - V[j][k])^2$ ;
14 end
15  $\Delta(i, j) = \sqrt{\lambda} \times \sqrt{d}$ ; // [eqn. 10]
16 write tuple  $\langle i, j, \Delta(i, j) \rangle$  into  $D$ ;
17 end

```

Algorithm 2: MAGSim (D)

```

// generating similarity graph corresponding to
a multi-attributed graph
Input : A CSV file  $D$  containing vertex-pairs along with
distance output by the MAGDist.
Output: A CSV file  $G_s$  containing edge-listed similarity
graph.
1  $d_{max} \leftarrow \text{getMaxDistance}(D)$ ;
2 for each tuple  $\langle i, j, \Delta(i, j) \rangle$  in  $D$  do
3 |  $\text{sim}(i, j) = 1 - \frac{\Delta(i, j)}{d_{max}}$  write tuple  $\langle i, j, \text{sim}(i, j) \rangle$  into  $G_s$ 
| ;
4 end

```

Example: Figure 2 presents a simpler case of multi-attributed graph having four vertices in which each vertex is represented as a two dimensional feature vector and each edge is represented as an one dimensional vector. In case, there is no direct edge between a pair of vertices (e.g., v_1 and v_3), the corresponding edge vector is a zero vector. If we simply calculate the Euclidean distance between the vertex-pairs, then the distance between the vertex-pairs (v_1, v_2) , (v_2, v_3) , (v_3, v_4) and (v_4, v_1) is same (10 unit), whereas the distance calculated by the proposed *MAGDist* differs, based on the weight of the edges between them. Similarly, the Euclidean distance between the vertex-pairs (v_1, v_3) and (v_2, v_4) are same ($10\sqrt{2}$), but the distance values calculated using *MAGDist* are different. The distance values between each vertex-pairs of the multi-attributed graph calculated using *MAGDist* for $\gamma = 1$ and $\gamma = 2$ are shown in D_1 and D_2 matrices, respectively of figure 2. It can be observed that giving more weightage to edges by increasing the value of γ reduces the distance between the respective vertex-pairs. Figure 3 presents a multi-attributed graph in which both vertex and edge labels are multi-dimensional vectors. For example, the edge vector corresponding to the edge connecting v_2 and v_3 is $\vec{e}(2, 3) = (0.36, 0.64)^T$. If we simply calculate the Euclidean distance between the vertex-pairs, then the distance between the vertex-pairs (v_1, v_2) , (v_2, v_3) , (v_3, v_4) and (v_4, v_1) is same (10 unit), whereas the distance calculated by the proposed *MAGDist* differs, based on the weight of the edges between them. The distance values between each vertex-pairs of the multi-attributed graph calculated using *MAGDist* for $\gamma = 1$ and $\gamma = 2$ are shown in D_1 and D_2 matrices, respectively of figure 3. It can be observed from these two distance matrices too that giving more weightage to edges by increasing the value of γ reduces the distance between the respective vertex-pairs.

4 EXPERIMENTAL SETUP AND RESULTS

To establish the efficacy of the proposed *MAGDist* distance measure, we have considered the well-known Iris data set¹, which contains total 150 instances of three different categories of Iris flower; 50 instances of each category. The Iris data set can be represented as a 150×5 data matrix, wherein each row represents an Iris flower, first four columns represent four different attribute values in centimetre, and the 5th column represents class labels (categories) of

¹<http://archive.ics.uci.edu/ml/datasets/Iris>

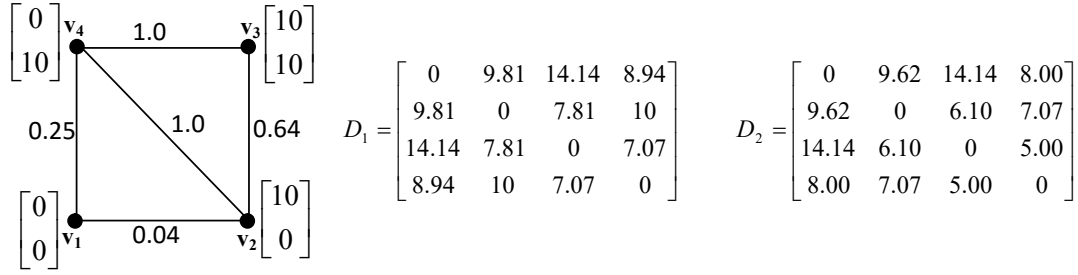


Figure 2: A multi-attributed graph with vertices as multi-dimensional vectors, and distance matrices D_1, D_2 calculated using *MAGDist* algorithm for $\gamma = 1$ and $\gamma = 2$, respectively

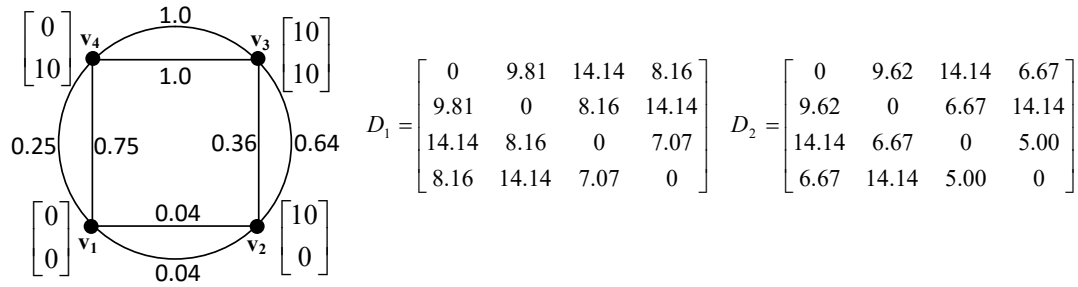


Figure 3: A multi-attributed graph with both vertices and edges as multi-dimensional vectors, and distance matrices D_1, D_2 calculated using *MAGDist* algorithm for $\gamma = 1$ and $\gamma = 2$, respectively

Table 1: A partial view of the Iris data set

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Setosa
4.9	3.0	1.4	0.2	Setosa
4.7	3.2	1.3	0.2	Setosa
4.6	3.1	1.5	0.2	Setosa
5.0	3.6	1.4	0.2	Setosa
...
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4.0	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5.7	2.8	4.5	1.3	versicolor
...
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica

the Iris flower as *Setosa*, *Virginica*, or *Versicolour*. Table 1 shows a partial snapshot of the Iris data set. We model Iris data set as a multi-attributed similarity graph in which each vertex $v \in \mathbb{R}^4$ is a 4-dimensional vector representing a particular instance of the Iris flower. The edge (similarity) between a vertex-pair (u, v) is determined using equation 14 on the basis of the Gaussian kernel value

defined in equation 15, where $\sigma = 1$ is a constant value [23].

$$e(u, v) = \begin{cases} \kappa^G(u, v) & \text{if } \kappa^G(u, v) \geq 0.55 \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

$$\kappa^G(u, v) = e^{-\frac{\|u-v\|^2}{2\sigma^2}} \quad (15)$$

The resulting multi-attributed Iris similarity graph is shown in figure 4 (termed hereafter as G_1 for rest of the paper), which contains 150 vertices and 2957 edges. In this graph, the instances of *Setosa*, *Virginica*, or *Versicolour* are shown using *triangles*, *squares*, and *circles*, respectively. Although $\kappa^G(u, u) = 1.0$ for all vertices, we haven't shown self-loops in this graph. The proposed *MAGDist* algorithm is applied over G_1 to calculate distance between all vertex-pairs, and finally *MAGSim* algorithm is applied to generate Iris similarity graph (termed hereafter as G_2 for rest of the paper), which is shown in figure 5. In [23], the authors have also used Gaussian kernel followed by the concept of nearest-neighbours to generate Iris similarity graph (termed hereafter as G_3 for rest of the paper) and applied Markov Clustering (MCL) to classify the instances of the Iris data into three different categories. Therefore, in order to verify the significance of our *MAGDist* and *MAGSim* algorithms, we have also applied the MCL over the Iris similarity graphs G_1 and G_2 , and present a comparative analysis of all clustering results. Figures 6 and 7 present the clustering results after applying MCL over the Iris similarity graphs G_1 and G_2 , respectively. Table 2 presents the contingency table for the discovered clusters versus true Iris types from all three different Iris similarity graphs. It can be observed from this table that, in case of G_2 , only six instances of iris-versicolor

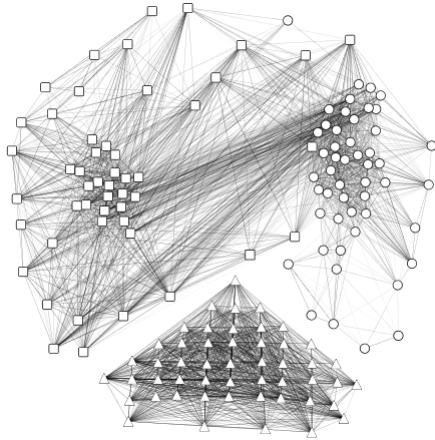


Figure 4: G_1 : Iris data graph modelled as a multi-attributed graph in which vertices are 4-dimensional real vectors and edges are the Gaussian similarity (≥ 0.55) between the respective vertices

are wrongly grouped with iris-virginica in C_2 and three instances of iris-virginica are wrongly grouped with iris-versicolor in C_3 ; whereas in G_1 forty iris-versicolor instances are wrongly grouped with iris-virginica in C_2 , and in G_3 , one instance of iris-versicolor is wrongly grouped with iris-setosa in C_1 and fourteen instances of iris-virginica are wrongly grouped with iris-versicolor in C_3 .

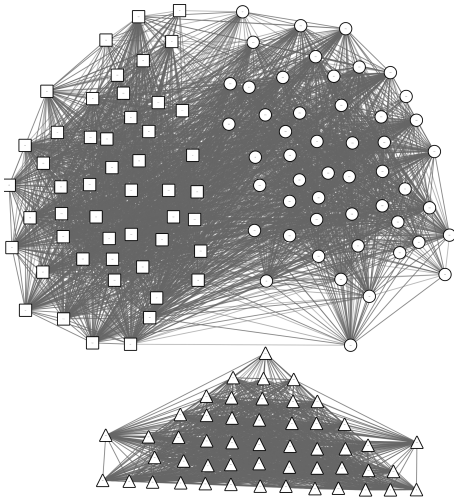


Figure 5: G_2 : Iris data graph modelled as a multi-attributed graph in which vertices are 4-dimensional real vectors and edges are MAGSim similarity (≥ 0.80) calculated using equation 13

We have also analyzed the significance of different similarity graph generation methods in terms of *True Positive Rate (TPR)* and *False Positive Rates (FPR)* that are defined using equations 16 and 17, respectively. In these equations, TP is the *True Positives*, representing the number of correctly classified positive instances, FP is

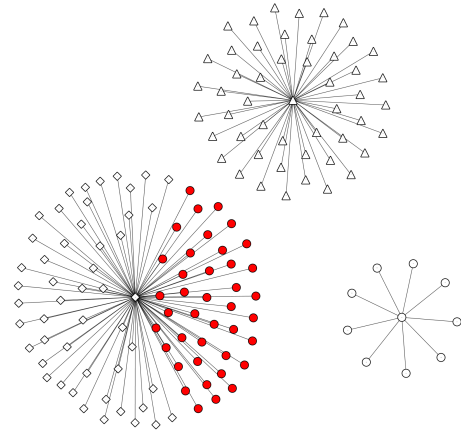


Figure 6: Clustering results after applying MCL over the Iris data graph G_1

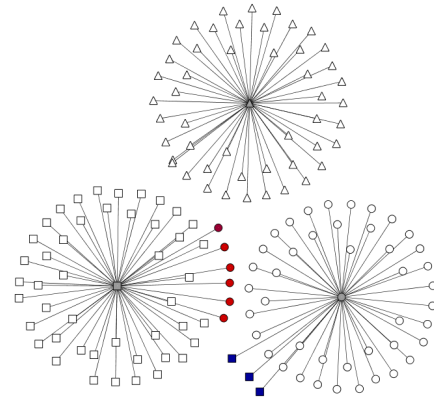


Figure 7: Clustering results after applying MCL over the Iris data graph G_2

Table 2: Contingency table for MCL clusters from similarity graphs generated by three different methods

Clusters	G_1			G_2			G_3 [23]		
	Set	Vir	Ver	Set	Vir	Ver	Set	Vir	Ver
C_1 (triangle)	50	0	0	50	0	0	50	0	1
C_2 (square)	0	50	40	0	47	6	0	36	0
C_3 (circle)	0	0	10	0	3	44	0	14	49

the *False Positives*, representing the number of wrongly classified negative instances, and P and N represent the total number of positive and negative instances, respectively in the data set. Table 3 presents *TPR* and *FPR* values for all three different similarity graphs, showing best results for G_2 , which has been generated using our proposed *MAGDist* and *MAGSim* algorithms.

$$TPR = \frac{TP}{P} \tag{16}$$

$$FPR = \frac{FP}{N} \tag{17}$$

Table 3: Performance comparison of three different methods on Iris data set

Clusters	G_1		G_2		G_3 [23]	
	TPR	FPR	TPR	FPR	TPR	FPR
C_1 (triangle)	1.00	0.00	1.00	0.00	1.00	0.01
C_2 (square)	1.00	0.40	0.94	0.06	0.72	0.00
C_3 (circle)	0.20	0.00	0.88	0.03	0.98	0.14
Average	0.73	0.13	0.94	0.03	0.90	0.05

4.1 Evaluation Results on Twitter Data Set

In order to illustrate the application of the proposed distance measure over a real multi-attributed graph, we have considered a Twitter data set of 300 tweets, comprising equal number of tweets related to three different events – *NoteBandi (NTB)*, *RyanInternationalSchool (RIS)*, and *Rohingya (ROH)*. The tweets are modelled as a multi-attributed graph, wherein each vertex represents a tweet as an 110-dimensional binary vector based on the top-110 key-terms identified using *tf-idf*, and two different edges exist between a vertex-pair – one representing the degree of *Hashtags* overlap and the other representing the *tweet-time* overlap. The similarity graph is generated using *MAGSim* algorithm, and shown in figure 8 wherein the instances of *NoteBandi*, *RyanInternationalSchool*, and *Rohingya* are represented using *squares*, *triangles*, and *circles*, respectively. Finally, *MCL* is applied over the similarity graph to group the tweets into different clusters shown in figure 9. The evaluation of the obtained clusters is given in table 4. It can be seen from this table that only five instances of *RyanInternationalSchool* are wrongly clustered with *Rohingya* in C_3 .

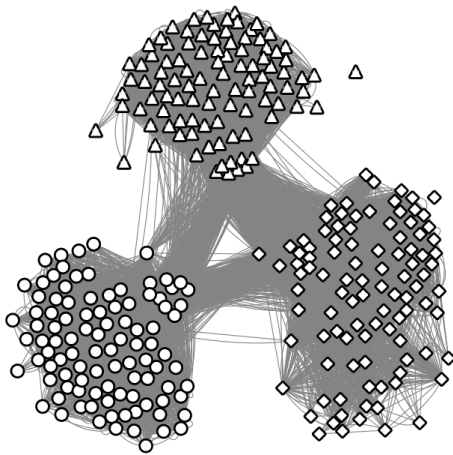


Figure 8: Similarity graph generated from Twitter data set (only edges having similarity value > 0.5 are shown)

5 CONCLUSION AND FUTURE WORKS

In this paper, we have proposed a novel weighted distance measure based on weighted Euclidean norm that can be used to calculate the distance between vertex-pairs of a multi-attributed graph containing multi-labelled vertices and multiple edges between a single

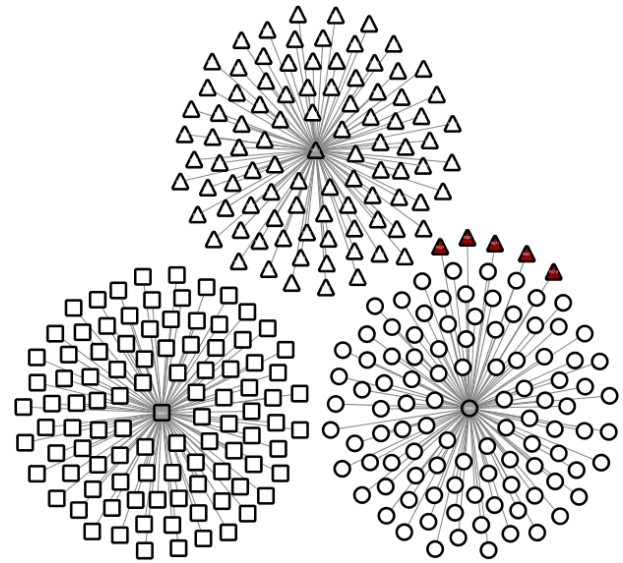


Figure 9: Clustering results obtained by applying MCL over the similarity graph of the Twitter data set

Table 4: Evaluation results of the proposed method on Twitter data set

Clusters	Contingency Table			Evaluation Results	
	NTB	RIS	ROH	TPR	FPR
C_1 (triangle)	0	95	0	1.00	0.00
C_2 (square)	100	0	0	1.00	0.00
C_3 (circle)	0	5	100	1.00	0.025
Average				1.00	0.008

vertex-pair. The proposed distance measure considers both vertex and edge weight-vectors, and it is flexible enough to assign different weightage to different edges and scale the overall edge-weight while computing the weighted distance between a vertex-pair. We have also proposed a *MAGDist* algorithm that reads the lists of vertex and edge vectors as two separate CSV files and calculates the distance between each vertex-pairs using the proposed weighted distance measure. Finally, we have proposed a multi-attributed similarity graph generation algorithm, *MAGSim*, which reads the output produced by the *MAGDist* algorithm and generates a similarity graph, which can be used by the existing classification and clustering algorithms for various analysis tasks. Since the proposed *MAGDist* algorithm reads multi-attributed graph as CSV files containing vertex and edge vectors, it can be scaled to handle large complex graphs (aka big graphs). Applying proposed distance measure and algorithms on (research articles) citation networks and online social networks to analyze them at different levels of granularity is one of the future directions of work.

REFERENCES

[1] Muhammad Abulaish and Sajid Yousuf Bhat. 2015. Classifier Ensembles using Structural Features for Spammer Detection in Online Social Networks. *Foundations of Computing and Decision Sciences* 40, 2 (2015), 89–105.

- [2] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. 1998. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proceedings of the ACM SIGMOD international conference on Management of data*. Seattle, Washington, USA, 94–105.
- [3] Faraz Ahmed and Muhammad Abulaish. 2013. A Generic Statistical Approach for Spam Detection in Online Social Networks. *Computer Communications* 36, 10–11 (2013), 1120–1129.
- [4] Sajid Yousuf Bhat and Muhammad Abulaish. 2013. Analysis and Mining of Online Social Networks - Emerging Trends and Challenges. *WIREs Data Mining and Knowledge Discovery* 3, 6 (2013), 408–444.
- [5] Sajid Yousuf Bhat and Muhammad Abulaish. 2014. HOCTracker: Tracking the Evolution of Hierarchical and Overlapping Communities in Dynamic Social Networks. *IEEE Transactions on Knowledge and Data Engineering* 27, 4 (2014), 1019–1032. DOI: <http://dx.doi.org/10.1109/TKDE.2014.2349918>
- [6] Sajid Yousuf Bhat and Muhammad Abulaish. 2014. Using communities against deception in online social networks. *Computer Fraud & Security* 2014, 2 (2014), 8–16.
- [7] Sajid Yousuf Bhat and Muhammad Abulaish. 2015. OCMiner: A Density-Based Overlapping Community Detection Method for Social Networks. *Intelligent Data Analysis* 19, 4 (2015), 1–31.
- [8] Sajid Yousuf Bhat and Muhammad Abulaish. 2017. A Unified Framework for Community Structure Analysis in Dynamic Social Networks. In *Hybrid Intelligence for Social Networks*, Hema Banati, Siddhartha Bhattacharyya, Ashish Mani, and Mario Koppen (Eds.). Springer, 1–21.
- [9] Sajid Y. Bhat, Muhammad Abulaish, and Abdulrahman A. Mirza. 2014. Spammer Classification using Ensemble Methods over Structural Social Network Features. In *Proceedings of the 14th IEEE/WIC/ACM International Conference on Web Intelligence (WIÅ14)*. Warsaw, Poland, 11–14.
- [10] Mohd Fazil and Muhammad Abulaish. 2017. Identifying Active, Reactive, and Inactive Targets of Socialbots in Twitter. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WIÅ17), Leipzig, Germany*. ACM, 573–580.
- [11] Mohd Fazil and Muhammad Abulaish. 2017. Why a Socialbot is Effective in Twitter? A Statistical Insight. In *Proceedings of the 9th International Conference on Communication Systems and Networks (COMSNETS), Social Networking Workshop, Bengaluru, India*. 562–567.
- [12] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. 1998. Inferring Web Communities from Link Topology. In *Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia*. ACM, New York, USA, 225–234.
- [13] M. Girvan and M. E. J. Newman. 2002. Community structure in social and biological networks. In *Proceedings of the National Academy of Sciences*. USA, 8271–8276.
- [14] Natallia Katenka and Eric D. Kolaczyk. 2011. Multi-Attribute Networks and the Impact of Partial Information on Inference and Characterization. *The Annals of Applied Statistics* 6, 3 (2011), 1068–1094.
- [15] Mladen Kolar, Han Liu, and Eric P. Xing. 2014. Graph Estimation From Multi-Attribute Data. *Journal of Machine Learning Research* 15 (2014), 1713–1750. <http://jmlr.org/papers/v15/kolar14a.html>
- [16] M. E. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69, 026113 (Feb 2004).
- [17] Raymond T. Ng and Jiawei Han. 1994. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*. Santiago, Chile, 144–155.
- [18] Peter J. Olver. 2008. *Numerical Analysis Lecture Note*. Retrieved on 18.03.2017 from http://www-users.math.umn.edu/~olver/num_/lnn.pdf.
- [19] S. E. Schaeffer. 2007. Graph clustering. *Computer Science Review* 1, 1 (2007), 27–64. DOI: <http://dx.doi.org/10.1016/j.cosrev.2007.05.001>
- [20] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. 2008. Efficient Aggregation for Graph Summarization. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Vancouver, Canada, 567–580.
- [21] E.W. Weisstein. 2002. *Vector Norm*. Wolfram MathWorld. Retrieved on 18.03.2017 from <http://mathworld.wolfram.com/VectorNorm.html>.
- [22] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A. J. Schweiger. 2007. SCAN: A Structural Clustering Algorithm for Networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Jose, California, USA, 824–833.
- [23] Mohammed J. Zaki and Wagner Meira Jr. 2014. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, New York, USA.
- [24] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph Clustering Based on Structural/Attribute Similarities. *Vldb Endowment* 2, 1 (Aug 2009), 718–729. DOI: <http://dx.doi.org/10.14778/1687627.1687709>