Graph Embedding on Spheres and its Application to Visualization of Information Diffusion Data

Kazumi Saito
University of Shizuoka
Shizuoka, Japan
k-saito@u-shizuoka-ken.ac.jp

Masahiro Kimura
Ryukoku University
Otsu, Japan
kimura@rins.ryukoku.ac.jp

Kouzou Ohara
Aoyama Gakuin University
Kanagawa, Japan
ohara@it.aoyama.ac.jp

Hiroshi Motoda
Osaka University
Osaka, Japan
motoda@ar.sanken.osaka-u.ac.jp

ABSTRACT

We address the problem of visualizing structure of undirected graphs that have a value associated with each node into a K-dimensional Euclidean space in such a way that 1) the length of the point vector in this space is equal to the value assigned to the node and 2) nodes that are connected are placed as close as possible to each other in the space and nodes not connected are placed as far apart as possible from each other. The problem is reduced to K-dimensional spherical embedding with a proper objective function. The existing spherical embedding method can handle only a bipartite graph and cannot be used for this purpose. The other graph embedding methods, e.g., multi-dimensional scaling, spring force embedding methods, etc., cannot handle the value constraint and thus are not applicable, either. We propose a very efficient algorithm based on a power iteration that employs the double-centering operations. We apply the method to visualize the information diffusion process over a social network by assigning the node activation time to the node value, and compare the results with the other visualization methods. The results applied to four real world networks indicate that the proposed method can visualize the diffusion dynamics which the other methods cannot and the role of important nodes, e.g. mediator, more naturally than the other methods.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Parameter learning

Keywords

graph embedding, visualization, information diffusion

1. INTRODUCTION

Complex network is hard to understand. Visualization can help, but in reality it is not self-evident whether there exists a good general visualization scheme that satisfies most of our needs. Especially if we want to visualize the dynamics taking place over a network, the only solution seems to use animation over time, which is not what we are aiming at.

We consider the following problem: Visualize the structure of an undirected graph that has a value assigned to each node in a K-dimensional Euclidean space in such a way that 1) the length of the point vector in this space is equal to the node value and 2) nodes that are connected are placed as close as possible to each other in the space and nodes not connected are placed as far apart as possible from each other. The constraint 1) is unique to this method and brings more flexibility in visualization. In fact, this enables to visualize a dynamics mentioned in the beginning.

The need for visualization is so high that various graph embedding methods have already been proposed and are widely used. These include multi-dimensional scaling [15], spectral embedding [2], spring force embedding [4] and cross-entropy embedding [16]. All of them are applicable to undirected graphs. Spherical embedding [11, 3] that came a little later is designed to visualize bipartite graphs. Among these five, the first four cannot handle the constraint 1). The last one cannot apply to a general undirected graph. To our knowledge, there is no method that can directly handle our problem. Further, apart from the above problem, those that solve non-linear optimization problem by a power iteration, except [3], are extremely slow.

We show that the above visualization problem is reduced to spherical embedding that is formulated as a non-linear optimization problem which maximizes a certain objective function that involves an operation called "double-centering". The problem can be solved by a simple power iteration as is done in the above existing methods, but this is very inefficient. We propose a much more efficient algorithm making effective use of the sparsity of the adjacency matrix, which is true for most complex networks. We verify that the algorithm works as intended by applying it to the visualization of information diffusion process over a large social network by assigning the node activation time to the node value (detail in Section 4.2). The results obtained by four real world social networks confirm our conjecture. Time evolution of the diffusion process is easily visualized by the proposed method and in this process such nodes that have a role of mediating the diffusion are more easily identifiable than the other existing methods which cannot handle the diffusion dynamics.

This paper is organized as follows. We first describe the

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

problem framework of embedding undirected graphs into a low dimensional Euclidean space (2.1.), show a simple update method for solving the optimal solution (2.1), followed by the proposed efficient update method (2.3). Next we briefly compare the proposed method with four existing methods (3). We then explain how we apply the method to the visualization of information diffusion (4), and report the results (5). We conclude the paper by summarizing what has been achieved (7).

2. SPHERICAL GRAPH EMBEDDING

We describe the framework of embedding an undirectded graph $G = (V, E)$ without self-loops into a K-dimensional Euclidean space, where V and E ($\subset V \times V$) stand for the sets of all the nodes and links, respectively. For the sake of technical convenience, we identify the set of the nodes, V, by a series of positive integers, i.e., $V = \{1, \cdots, m, \cdots, M\}$. Here M is the number of the nodes in V , i.e., $|V| = M$. Then, we can define the $M \times M$ adjacency matrix $\mathbf{A} = \{a_{m,n}\}$ by setting $a_{m,n} = 1$ if $\{m, n\} \in E$; $a_{m,n} = 0$ otherwise. Note taht $a_{m,n} = a_{n,m}$ and $a_{m,m} = 0$. We denote the K-dimensional embedding position vectors by $\mathbf{x}_m$ for the node $m \in V$. Then we can construct the $K \times M$ matrix consisting of these position vectors, i.e., $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_M)$.

2.1 Problem Formulation

We first state the framework of our embedding problem intuitively: For a given undirectded graph $G = (V, E)$ and a set of values assigned to each node, denoted by $(r_1, \cdots, r_m, \cdots, r_M)$, we attempt to visualize the graph so that each pair of nodes with similar connection patterns is embedded as a pair of position vectors with similar directions, and each length of the embedded position vectors is set to the above value assigned to the node, i.e., $\|\mathbf{x}_m\| = r_m$ for each m, where $\|\mathbf{x}_m\|$ stands for the norm of the vector $\mathbf{x}_m$.

In order to more closely explain our embedding problem, we introduce the centering (Young-Householder transformation) matrix,

$$\mathbf{H}_M = \mathbf{I}_M - \frac{1}{M}\mathbf{1}_M\mathbf{1}_M^T, \tag{1}$$

where $\mathbf{I}_M$ stands for the $M \times M$ identity matrix, $\mathbf{1}_M$ is an M-dimensional vector whose elements are all one, and $\mathbf{1}^T$ means the transposition of the vector $\mathbf{1}$. Clearly, the mean vector of the resulting position vectors becomes $\mathbf{0}$ by the operations $\mathbf{X}\mathbf{H}_M$. Then, we consider the following double-centered matrix $\mathbf{B} = \{b_{m,n}\}$ that is calculated from the adjacency matrix $\mathbf{A}$.

$$\mathbf{B} = \mathbf{H}_M\mathbf{A}\mathbf{H}_M. \tag{2}$$

Note that the mean vectors of both the row and the column vectors of the matrix $\mathbf{B}$ become $\mathbf{0}$. On the other hand, for position vectors $\{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$, we can consider the similarity matrix $\mathbf{C} = \{c_{m,n}\}$, each element of which is defined by the following cosine similarity.

$$c_{m,n} = \frac{\mathbf{x}_m^T}{\|\mathbf{x}_m\|}\frac{\mathbf{x}_n}{\|\mathbf{x}_n\|}. \tag{3}$$

As the basic strategy of our graph embedding, we maximize the correlation between the the double-centered matrix $\mathbf{B}$ and the cosine similarity matrix $\mathbf{C}$ by adequately locating each position vector under the constraints $\|\mathbf{x}_m\| = r_m$. Namely, we can consider the following objective function with respect to the matrix $\mathbf{X}$ constructed from the position vectors.

$$\begin{aligned} J(\mathbf{X}) &= \sum_{m=1}^{M-1}\sum_{n=m+1}^{M} b_{m,n}c_{m,n} + \frac{1}{2}\sum_{m=1}^{M}\lambda_m(r_m^2 - \mathbf{x}_m^T\mathbf{x}_m) \\ &= \sum_{m=1}^{M-1}\sum_{n=m+1}^{M} b_{m,n}\frac{\mathbf{x}_m^T}{r_m}\frac{\mathbf{x}_n}{r_n} + \frac{1}{2}\sum_{m=1}^{M}\lambda_m(r_m^2 - \mathbf{x}_m^T\mathbf{x}_m) \end{aligned} \tag{4}$$

where $\{\lambda_m \mid m = 1, \cdots, M\}$ correspond to Lagrange multipliers for the constraints, i.e., $\mathbf{x}_m^T\mathbf{x}_m = r_m^2$ for $1 \le m \le M$. Intuitively, maximizing $J(\mathbf{X})$ pushes the pairs $\mathbf{x}_m$ and $\mathbf{x}_n$ to the same direction if they are connected and pushes them to the opposite direction if they are unconnected, and realizes the intended visualization.

Now, we consider a reparameterization of each position vector $\mathbf{x}_m$ by $\tilde{\mathbf{x}}_m = \mathbf{x}_m/r_m$, and set $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_M)^T$. Then, we can equivalently transform our objective function defined in Equation (4) as follows,

$$J(\tilde{\mathbf{X}}) = \sum_{m=1}^{M-1}\sum_{n=m+1}^{M} b_{m,n}\tilde{\mathbf{x}}_m^T\tilde{\mathbf{x}}_n + \frac{1}{2}\sum_{m=1}^{M}\mu_m(1 - \tilde{\mathbf{x}}_m^T\tilde{\mathbf{x}}_m), \tag{5}$$

where $\mu_m = \lambda_m/r_m^2$ for each m. Thus, maximizing Equation (4) is implemented by the following two steps: First, we calculate the position vector $\tilde{\mathbf{x}}_m$ for each node on the unit sphere (circle), so as to maximize Equation (5); Then, we can obtain the final position vectors just by rescaling them with respect to $(r_1, \cdots, r_M)$, i.e., $\mathbf{x}_m = r_m\tilde{\mathbf{x}}_m$ for each m. Thus we can regard our problem as a shperical graph embedding problem on the unit sphere. Hereafter, we simply denote $\tilde{\mathbf{x}}_m$ as $\mathbf{x}_m$ in order to avoid notational complication. Here we should emphasize that in our problem formalization, the directions of the embedded position vectors are determined independently from the values assigned to each node.

2.2 Simple Update Method

Now we consider maximizing $J(\mathbf{X})$ defined in Equation (5) by use of a coordinate strategy: We maximize $J(\mathbf{X})$ with respect to each position vector $\mathbf{x}_m$, by fixing the other position vectors. In order to optimally update each position vector $\mathbf{x}_m$, we consider the following gradient vector of the objective function $J(\mathbf{X})$ with respect to $\mathbf{x}_m$.

$$\frac{\partial J(\mathbf{X})}{\partial \mathbf{x}_m} = \sum_{n=1, n\neq m}^{M} b_{m,n}\mathbf{x}_n - \mu_m\mathbf{x}_m. \tag{6}$$

Thus, for the fixed vectors $\{\mathbf{x}_1, \cdots \mathbf{x}_M\} \setminus \mathbf{x}_m$, we obtain the optimal position vector $\mathbf{x}_m$ which maximizes the objective function $J(\mathbf{X})$ as follow:

$$\mathbf{x}_m = \frac{1}{\|\mathbf{f}_m\|}\mathbf{f}_m, \tag{7}$$

where

$$\mathbf{f}_m = \sum_{n=1, n\neq m}^{M} b_{m,n}\mathbf{x}_n = (\mathbf{X} - \mathbf{x}_m\mathbf{e}_m^T)\mathbf{B}\mathbf{e}_m. \tag{8}$$

Here $\mathbf{e}_m$ is an M-dimensional unit vector whose m-th element is 1, and the other elements are 0.

However, this simple iteration method requires the computational complexity of $O(MK)$ for updating each optimal

position vector according to Equation (8). In order to make better use of the sparsity of adjacency matrix which is frequently observed in most complex networks, we derive an efficient way of calculating Equation (8) in the succeeding subsection.

2.3 Efficient Update Method

We first focus on the following equivalent formula for calculating $\mathbf{f}_m$ in Equation (8).

$$\mathbf{f}_m = \mathbf{X}\mathbf{B}\mathbf{e}_m - (\mathbf{e}_m^T\mathbf{B}\mathbf{e}_m)\mathbf{x}_m. \tag{9}$$

Here we consider a degree vector defined by

$$\mathbf{d} = (d_1, \cdots, d_M)^T = \mathbf{A}\mathbf{1}_M, \tag{10}$$

and their average,

$$D = \frac{1}{M}\mathbf{1}_M^T\mathbf{d} = \frac{1}{M}\mathbf{1}_M^T\mathbf{A}\mathbf{1}_M. \tag{11}$$

Then, from the definition of double-centered matrix $\mathbf{B}$ given in Equation (2), we can calculate $\mathbf{B}\mathbf{e}_m$ as follows.

$$\begin{aligned}\mathbf{B}\mathbf{e}_m &= (\mathbf{I}_M - \frac{1}{M}\mathbf{1}_M\mathbf{1}_M^T)\mathbf{A}(\mathbf{I}_M - \frac{1}{M}\mathbf{1}_M\mathbf{1}_M^T)\mathbf{e}_m \\ &= \mathbf{A}\mathbf{e}_m + \frac{D-d_m}{M}\mathbf{1}_M - \frac{1}{M}\mathbf{d}. \end{aligned}\tag{12}$$

By noting that $\mathbf{e}_m^T\mathbf{A}\mathbf{e}_m = 0$ because of no self-loops, we obtain $\mathbf{e}_m^T\mathbf{B}\mathbf{e}_m$ as follows.

$$\mathbf{e}_m^T\mathbf{B}\mathbf{e}_m = \frac{D-2d_m}{M} \tag{13}$$

Now we define the average position vector $\boldsymbol{\phi}$ and the degree-weighted average position vector $\boldsymbol{\psi}$ by

$$\boldsymbol{\phi} = \frac{1}{M}\mathbf{X}\mathbf{1}_M, \quad \boldsymbol{\psi} = \frac{1}{M}\mathbf{X}\mathbf{d}, \tag{14}$$

respectively. Then by substituting Equations (12) and (13) into Equation (9), we can obtain the following.

$$\mathbf{f}_m = \sum_{n\in\Gamma(m)}\mathbf{x}_n + (D-d_m)\boldsymbol{\phi} - \boldsymbol{\psi} - \frac{D-2d_m}{M}\mathbf{x}_m, \tag{15}$$

where, $\Gamma(m)$ denotes a set of neighbour nodes of v, *i.e.*, those nodes that are connected to v. Thus by noting that both $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are K-dimensional vectors, and the average number of elements in $\Gamma(m)$ is D, i.e., $D =< |\Gamma(m)| >$, we can see that the average computational complexity of calculating $\mathbf{f}_m$ is reduced to $O(DK)$ from $O(MK)$ in average. As mentioned earlier, we can naturally assume $M \gg D$ for a wide variety of complex networks.

On the other hand, after updating the position vector $\mathbf{x}_m$, we need to update vectors $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ according to this change as well. For this purpose, after setting the updated vector $\mathbf{y}_m$ and the modification vector $\Delta\mathbf{x}_m$ by,

$$\mathbf{y}_m = \frac{1}{\|\mathbf{f}_m\|}\mathbf{f}_m, \quad \Delta\mathbf{x}_m = \mathbf{y}_m - \mathbf{x}_m, \tag{16}$$

we update the vectors $\boldsymbol{\phi}$, $\boldsymbol{\psi}$, and $\mathbf{x}_m$ as follows.

$$\boldsymbol{\phi} = \boldsymbol{\phi} + \frac{1}{M}\Delta\mathbf{x}_m, \quad \boldsymbol{\psi} = \boldsymbol{\psi} + \frac{d_m}{M}\Delta\mathbf{x}_m, \quad \mathbf{x}_m = \mathbf{y}_m. \tag{17}$$

Clearly, these updates can be done within the computational complexity of $O(K)$. Thus, we can see that the computational complexity of updating $\mathbf{x}_m$ is equal to $O(DK)$.

Below we summarize our spherical embedding algorithm proposed in this paper.

1. Initialize position vectors $\{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$ adequately; and calculate vectors $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ by Equation (14);

2. For each $m \in \{1, \cdots, M\}$, calculate $\mathbf{f}_m$ by Equation (15), set vectors $\mathbf{y}_m$ and $\Delta\mathbf{x}_m$ by Equation (16), and update vectors $\boldsymbol{\phi}$, $\boldsymbol{\psi}$, and $\mathbf{x}_m$ by Equation (17);

3. If $\max_m\{\|\partial J(\mathbf{X})/\partial\mathbf{x}_m\|\} < \epsilon$, output $\{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$ and terminate;

4. Return to the step 2.

Our proposed algorithm employs a power iteration as the basic framework, just like the HITS algorithm [8], which utilizes $\mathbf{A}$ and $\mathbf{A}^T$, does. However, the main differences are use of the double-centering operation by $\mathbf{H}_M$, and the constraints described by $\|\mathbf{x}_m\| = r_m$. Here note that the double-centering operation is also employed in the standard multidimensional scaling method [15].

Now we briefly mention the computational complexity of our algorithm. Clearly, the main computational complexity of one-iteration comes from the multiplication by the matrix $\mathbf{A}$ with position vectors $\mathbf{x}_m$, which is the most computationally intensive part and is proportional to the number of links in the undirected graph. Thus, the proposed algorithm is expected to work much faster especially for a sparse undirected graph. In fact, it has been well known that the PageRank algorithm [1] based on a power iteration works very fast for a large and sparse network [10] even without parallel distributed processing.

3. ALGORITHMIC COMPARISON WITH CONVENTIONAL METHODS

We compare the proposed method from algorithmic aspect with the four well known embedding methods: multidimensional scaling [15], spectral embedding [2], spring force embedding [4], and cross-entropy embedding [16]. Here the former two perform a power iteration with respect to either a double-centered distance matrix or a graph Laplacian matrix which is calculated from a given graph, while the latter two repeatedly move each position vector by using the Newton method in a framework of nonlinear optimization. Here note that the basic strategy of our method is a combination of the above basic strategy, i.e., our method performs a power iteration with respect to a double-centered adjacency matrix while repeatedly moving each position vector. However, recall that these existing methods cannot directly utilize the values associated with nodes. In what follows, we compare our method more closely with these existing methods.

Multi-dimensional scaling method [15] first calculates the distance matrix $\mathbf{G}$, and performs the double centering operation to the distance matrix. Mathematically it is formulated as minimizing Equation (18).

$$\mathcal{M}(\mathbf{X}) = \frac{1}{2}\sum_{k=1}^{K}\mathbf{z}_k^T(\mathbf{H}_M\mathbf{G}\mathbf{H}_M)\mathbf{z}_k, \tag{18}$$

where $\mathbf{z}_k = (x_{1,k}, \cdots, x_{M,k})^T$, and $\{\mathbf{z}_1, \cdots, \mathbf{z}_K\}$ need to be orthonormal vectors, i.e., $\mathbf{z}_k^T\mathbf{z}_k = 1$ and $\mathbf{z}_k^T\mathbf{z}_{k'} = 0$ if $k \neq k'$.

Spectral embedding method [2] tries to directly minimize distances between position vectors of connecting nodes. Math-

ematically it is formulated as minimizing Equation (19).

$$\mathcal{S}(\mathbf{X}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N} a_{m,n}(z_{k,m}-z_{k,n})^2 = \sum_{k=1}^{K}\mathbf{z}_k^T(\mathbf{D}-\mathbf{A})\mathbf{z}_k, \quad (19)$$

where $\mathbf{D}$ is a diagonal matrix each element of which is the degree of node (number of links). Note that $(\mathbf{D}-\mathbf{A})$ is referred to as a graph Laplacian matrix. Again, we set $\mathbf{z}_k = (x_{1,k},\cdots,x_{M,k})^T$, and $\{\mathbf{z}_1,\cdots,\mathbf{z}_K\}$ need to be orthonormal vectors, which excludes the trivial vector expressed as $\mathbf{z} \propto \mathbf{1}_M$.

Spring force embedding method [4] assumes that there is a hypothetical spring between each connected node pair and locates nodes such that the distance of each node pair is closest to its minimum path length at equilibrium. Mathematically it is formulated as minimizing Equation (20).

$$\mathcal{K}(\mathbf{X}) = \sum_{m=1}^{M-1}\sum_{n=m+1}^{M} \alpha_{m,n}(g_{m,n}-\|\mathbf{x}_m-\mathbf{x}_n\|)^2, \quad (20)$$

where $\alpha_{m,n}$ is a spring constant which is normally set to $1/(2g_{u,v}^2)$.

Cross-entropy embedding method [16] first defines a similarity $\rho(\mathbf{x}_m,\mathbf{x}_n)$ between the embedding positions x_m and x_n and uses the corresponding element $a_{m,n}$ of the adjacency matrix as a measure of distance between the node pair, and tries to minimize the total cross entropy between these two. Mathematically it is formulated as minimizing Equation (21).

$$\mathcal{C}(\mathbf{X}) = -\sum_{m=1}^{M-1}\sum_{n=m+1}^{M} (a_{m,n}\log\rho(\mathbf{x}_m,\mathbf{x}_n) + (1-a_{m,n})\log(1-\rho(\mathbf{x}_u,\mathbf{x}_v))). \quad (21)$$

Here, note that we used the function $\rho(\mathbf{x}_u,\mathbf{x}_v) = \exp(-\frac{1}{2}||\mathbf{x}_u-\mathbf{x}_v||^2)$ in our experiments.

The spectral embedding method is expected to work comparable to our method because these methods perform a power iteration on a sparse adjacency matrix. The multi-dimensional scaling method requires a substantially large computation time because it needs to perform a power iteration on a full distance matrix. Spring force embedding method and Cross-entropy embedding method both of which repeatedly move each position vector by using the Newton method, require an extremely large computation time before the final results are obtained.

4. APPLICATION TO VISUALIZATION OF INFORMATION DIFFUSION DATA

Our primary application of the proposed method is visualization of information diffusion process over a social network. We start with a brief description of the diffusion models we used and then describe how we visualize the diffusion data.

4.1 Information diffusion models

We focus on the IC (Independent Cascade) and the LT (Linear Threshold) models [5] as the representative models of information diffusion, and utilize their extended version that can cope with asynchronous time activation, AsIC (Asynchronous IC) and AsLT (Asynchronous LT) models [13, 14] in our experiments.

4.1.1 Asynchronous Independent Cascade Model

We first recall the definition of the IC model according to the work of [5], and then introduce the AsIC model. In the IC model, we specify a real value $p_{m,n}$ with $0 < p_{m,n} < 1$ for each link (m,n) in advance. Here $p_{m,n}$ is referred to as the *diffusion probability* through link (m,n). The diffusion process unfolds in discrete time-steps $t \geq 0$, and proceeds from a given initial active set S in the following way. When a node m becomes active at time-step t, it is given a single chance to activate each currently inactive child node n, and succeeds with probability $p_{m,n}$. If m succeeds, then n will become active at time-step $t+1$. If multiple parent nodes of n become active at time-step t, then their activation attempts are sequenced in an arbitrary order, but all performed at time-step t. Whether or not m succeeds, it cannot make any further attempts to activate n in subsequent rounds. The process terminates if no more activations are possible.

In the AsIC model, we specify real values $r_{m,n}$ with $r_{m,n} > 0$ in advance for each link $(m,n) \in E$ in addition to $p_{m,n}$, where $r_{m,n}$ is referred to as the *time-delay parameter* through link (m,n). The diffusion process unfolds in continuous-time t, and proceeds from a given initial active set S in the following way. Suppose that a node m becomes active at time t. Then, m is given a single chance to activate each currently inactive child node n. We choose a delay-time δ from the exponential distribution[1] with parameter $r_{m,n}$. If n has not been activated before time $t+\delta$, then m attempts to activate n, and succeeds with probability $p_{m,n}$. If m succeeds, then n will become active at time $t+\delta$. Said differently, whichever parent m that succeeds in satisfying the activation condition and for which the activation time is the earliest considering the time delay associated with each link can actually activate the node. Under the continuous time framework, it is unlikely that n is activated simultaneously by its multiple parent nodes exactly at time $t+\delta$. So we ignore this possibility. Whether or not m succeeds, it cannot make any further attempts to activate n in subsequent rounds. The process terminates if no more activations are possible.

4.1.2 Asynchronous Linear Threshold Model

Same as the above, we first recall the LT model. In this model, for every node $n \in V$, we specify a *weight* $(q_{m,n} > 0)$ from its parent node m in advance such that

$$\sum_{m\in B(n)} q_{m,n} \leq 1.$$

The diffusion process from a given initial active set S proceeds according to the following randomized rule. First, for any node $n \in V$, a *threshold* θ_n is chosen uniformly at random from the interval $[0,1]$. At time-step t, an inactive node n is influenced by each of its active parent nodes, m, according to weight $q_{m,n}$. If the total weight from active parent nodes of n is no less than θ_n, that is,

$$\sum_{m\in B_t(n)} q_{m,n} \geq \theta_n,$$

[1]Similar formulation can be derived for other distributions such as power-law and Weibull.

then n will become active at time-step $t+1$. Here, $B_t(n)$ stands for the set of all the parent nodes of n that are active at time-step t. The process terminates if no more activations are possible.

The AsLT model is defined in a similar way to the AsIC. In the AsLT model, in addition to the weight set $\{q_{m,n}\}$, we specify real values $r_{m,n}$ with $r_{m,n} > 0$ in advance for each link (m,n). Same as for AsIC, we refer to $r_{m,n}$ as the *time-delay parameter* through link (m,n). The diffusion process unfolds in continuous-time t, and proceeds from a given initial active set S in the following way. Each active parent m of the node n exerts its effect on n with the time delay δ drawn from the exponential distribution with the delay parameter $r_{m,n}$. Suppose that the accumulated weight from the active parents of node n has become no less than θ_n at time t for the first time. Then, the node n becomes active at t without any delay and exerts its effect on its child with a delay associated with its link. This process is repeated until no more activations are possible.

4.2 Visualization Method

Let $R = \{(m,t_m),(n,t_n),\cdots\}$ be an information diffusion result over an undirected $G=(V,E)$, where (n,t_n) is a pair of an activated node and its activation time. We set the initial activation time to 0. From the set of nodes that appear in R, i.e., $V' = \{n \mid (n,t_n) \in R\}$, we obtain an induced subgraph $G' = (V', E')$. Here, we regard t_n as n's associated value for $n \in V'$. If $m \in V'$, $n \in V'$, $(m,n) \in E'$, and $t_m < t_n$, the direction of information diffusion is limited to from node m to n. Namely, a directed acyclic graph (DAG) is constructed from the information diffusion result R. Although our embedding method is designed for undirected graph, we can interpret that the diffusion of information takes over from the origin to the periphery by setting the radius of node n to t_n. The major reason why we restricted the graph we handle to undirected graph is to maintain clear meaning of the objective function we are trying to maximize. Alternatively, we can start with a directed graph and obtain a directed induced subgraph. Then we reinterpret it as an undirected subgraph, and apply the above discussion.

5. EXPERIMENTAL EVALUATION

5.1 Datasets

We generated diffusion results using both the AsIC and the AsLT models for four large real social networks. They are all bidirectionally connected networks. The first one is a trackback network of Japanese blogs used in [7]. It has $12,047$ nodes and $79,920$ directed links (the blog network). The second one is a coauthorship network used in [12], which has $12,357$ nodes and $38,896$ directed links (the coauthorship network). The third one is a network derived from the Enron Email Dataset [9] by extracting the senders and the recipients and linking those that had bidirectional communications. It has $4,254$ nodes and $44,314$ directed links (the Enron network). The fourth one is a network of people that was derived from the "list of people" within Japanese Wikipedia, used in [6], and has $9,481$ nodes and $245,044$ directed links (the Wikipedia network).

5.2 Experimental Results

We visualized the information diffusion result in 2-dimensional Euclidean space, i.e., $K=2$, and compared the results of the proposed method with the other four existing methods. The initial active node was chosen to be the most influential node for each diffusion model. The location of this node is the origin of the visualization plane for the proposed method, but the location of the same node for the other methods is not controllable and determined by the algorithm of each method. The proposed method has the time information. A family of blue dotted circles of different radii centered at the origin indicates the activation times, where the radius t of each blue dotted circle corresponds to the actual time t. For all the visualization methods, red points and green lines are used to display the activated nodes and their links, respectively. It is noted that we are visualizing from the observed data, meaning that we don't know the parent which activated its child if there is more than one active parent. Thus, all the links between the activate parents and their active children are displayed.

Due to the space limitation, we only show parts of the results. Figure 1 shows the visualization result of information diffusion for the AsIC model over the Blog network using the proposed method, where the thick black circle indicate the initial active node. It is clear that the proposed method have the following properties:

1. Given two active nodes, we can easily see which one became active earlier.
2. Given an active node, we can easily identify its parents that could activate it (but we cannot identify it if there are multiple active parents by the reason mentioned above).

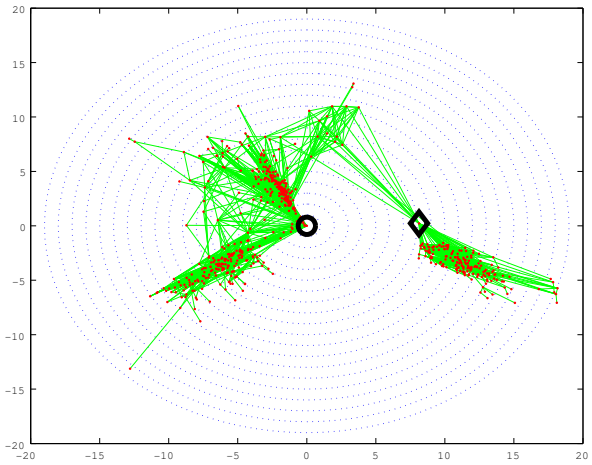

Figure 1: Visualization result of proposed method for the Blog network (AsIC model).

We can observe that in general *super-mediators*, i.e., those nodes that play an important role in passing the information to other nodes, are easily identified by the proposed method. In Figure 1, the thick black diamond node can naturally be interpreted as a super-mediator. The same node is also displayed as thick black diamonds in Figures 2, 3, 4 and 5. We notice that the multi-dimensional scaling and the spring force embedding methods are also good to find super-mediators, while it is more difficult to find them for the spectral embedding and the cross-entropy embedding methods. Note that the multi-dimensional scaling and the spring force

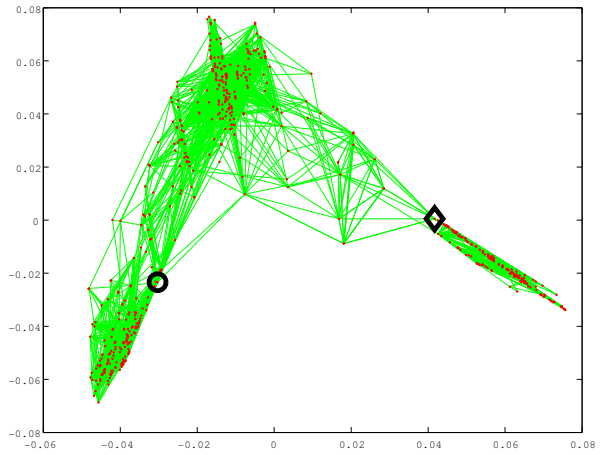

Figure 2: Visualization result of multi-dimensional scaling for the Blog network (AsIC model).

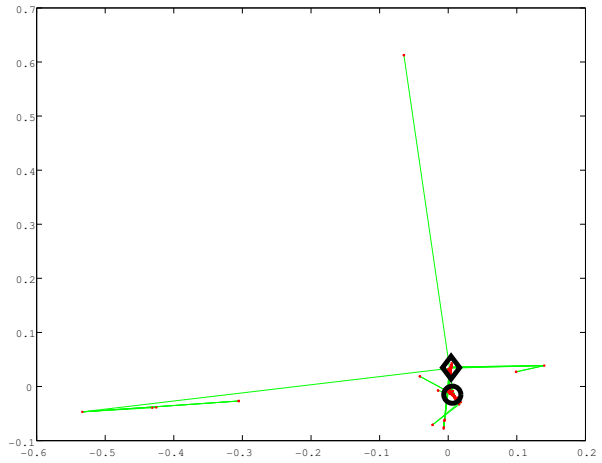

Figure 3: Visualization result of spectral embedding for the Blog network (AsIC model).

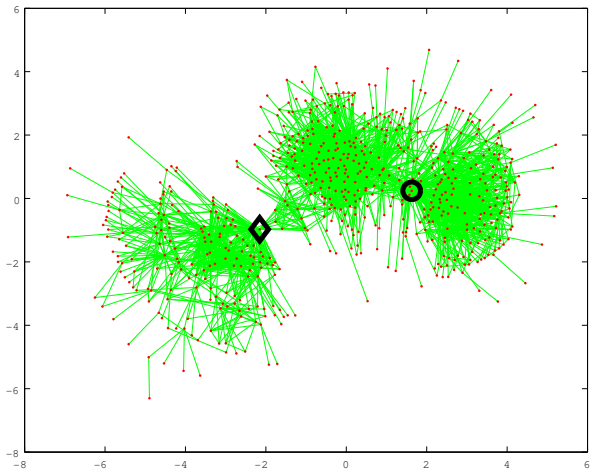

Figure 4: Visualization result of spring force embedding for the Blog network (AsIC model).

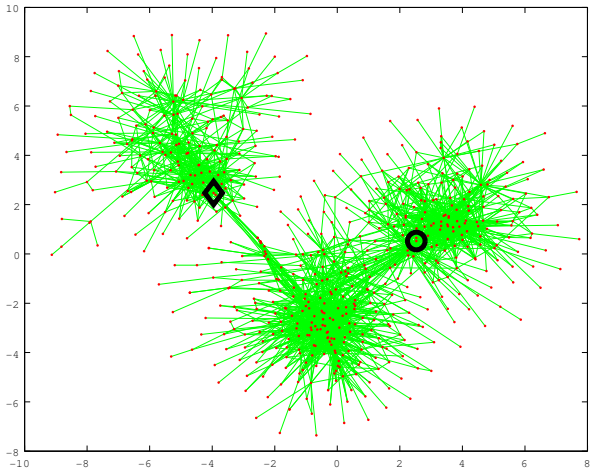

Figure 5: Visualization result of cross-entropy embedding for the Blog network (AsIC model).

embedding methods are based on graph distance matrix $\mathbf{G}$, and the spectral embedding and the cross-entropy embedding methods are based on graph adjacency matrix $\mathbf{A}$. For the $\mathbf{G}$-based methods, the distance from the initial active node (thick black circle) to an active node v in the visualization plane can be correlated with the time if the node v is an active node. Thus, we can consider that such methods have a possibility of finding super-mediators. However, we see from Figures 1 to 5 that the proposed method better identifies a super-mediator than the multi-dimensional scaling and the spring force embedding methods.

Figure 6 shows the visualization result of information diffusion for the AsLT model over the Blog network. Compared with the visualization result for the AsIC model, we observe that links are mostly outward directed and only small links are in circumferential direction. We consider that this observation comes from a characteristic difference between the AsIC and AsLT models. Especially, in case of the AsLT model, when a parent node becomes active, only its low degree child nodes are likely to be activated. Our proposed method will locate these child nodes to similar directions because their connectivity patterns are necessarily close. We consider that this fact partly explains the difference between the visualization results of Figure 1 and 6.

Figures 7 and 8 respectively show the visualization results of information diffusion for the AsIC and AsLT models over the Wikipedia network. We can also see from these figures that the proposed method is promising for identifying influential super-mediators and exploring the characteristic differences between the two information diffusion models. As mentioned earlier, the visualization results over the coauthorship and Enron networks are omitted due to the space limitation, but it is confirmed that we obtained similar results.

Last but not least, we evaluated our proposed method only in the case of two-dimensional embedding for our visualization purpose, but this does not mean that it is limited to two-dimensional embedding. It is quite easy to extend it to the general K-dimension embedding. We plan to evaluate our method as a powerful technique for both dimensional reduction and clustering as a future work.

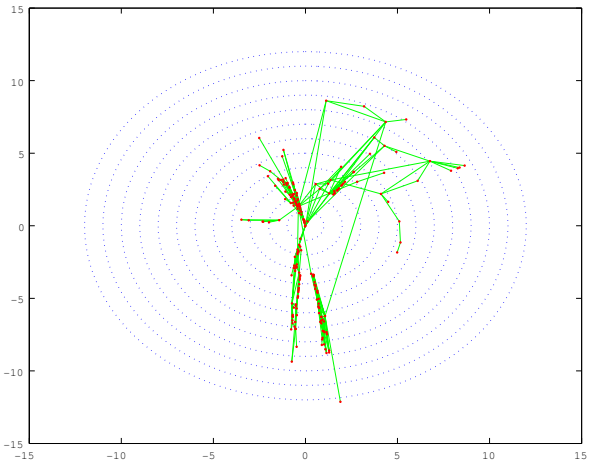

Figure 6: Visualization result of proposed method for the Blog network (AsLT model).

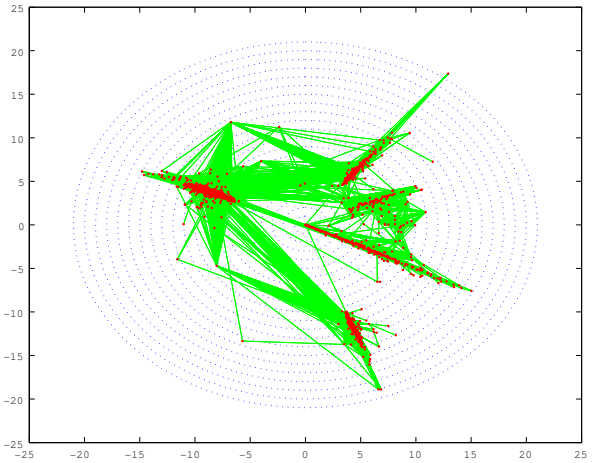

Figure 7: Visualization result of proposed method for the Wikipedia network (AsIC model).

6. DISCUSSION

One of the unique features of the proposed method is that we deal with the graph that has a value to each node, and the visualization takes account of the node value. The application to information diffusion involves time evolution and assigning the time the node gets activated to the node value works nicely to allow the diffusion starts at the origin always. On the contrary, all the other existing methods, when applied to the same visualization problem, generates a graph where the starting point of the diffusion is determined by the algorithm. Thus, if we want to visualize multiple results of diffusion sequences each starting from the same node, the starting node in each visualization is placed in a different location. Thus, the above feature is one of the advantages of the proposed method.

7. CONCLUSION

We addressed the problem of visualizing structure of a undirected graph that has a value associated with each node into a K-dimensional Euclidean space in such a way that 1) the length of the point vector in this space is equal to the value assigned to the node and 2) nodes that are connected are placed as close as possible to each other in the space and nodes not connected are placed as far apart as possible from each other. We showed that this visualization problem is reduced to spherical embedding that is formulated as a non-linear optimization problem for which a certain objective function to be maximized is defined. We proposed a very efficient algorithm based on a power iteration that employs double-centering operations. To validate the effectiveness of the proposed method, we applied it to visualize the information diffusion process over a social network by assigning the node activation time to the node value. We used the result of information diffusion obtained by two different diffusion models (AsIC and AsLT models) for four real world networks, and compared the proposed method with the multi-dimensional scaling, the spring force embedding, the spectral embedding and the cross-entropy embedding methods. We first confirmed that the proposed method can visualize time evolution of the diffusion process in an more intuitively understandable manner. We also confirmed that the proposed method have the following properties: 1) given two active nodes, we can easily see which one became active earlier, and 2) given an active node, we can easily identify its parents that could activate it (note that we are visualizing from the observed diffusion data, meaning that we don't know the parent which activated its child if there is more than one active parent.) Furthermore, we experimentally showed that the proposed method can better identify *super-mediators*, i.e., those nodes that play an important role in passing the information to other nodes, than the other four methods.

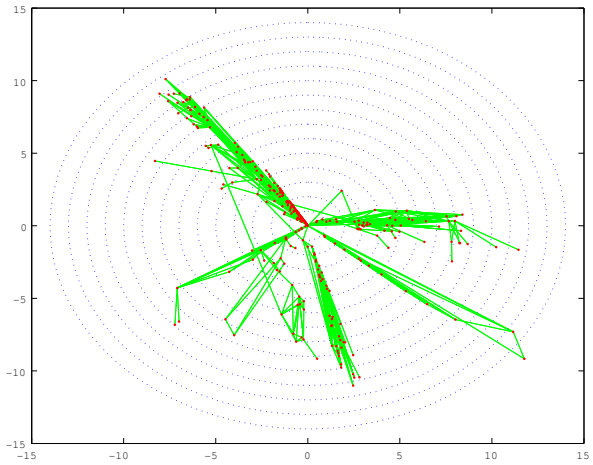

Figure 8: Visualization result of proposed method for the Wikipedia network (AsLT model).

8. ACKNOWLEDGMENTS

This work was partly supported by Asian Office of Aerospace Research and Development, Air Force Office of Scientific Research under Grant No. AOARD-11-4111, JSPS Grant-in-Aid for Scientific Research (C) (No. 23500312), and Toyota Central R&D Labs., Inc. (No. E11114).

9. REFERENCES

[1] S. Brin and L.Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.

[2] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, New York, 1997.

[3] T. Fushimi, Y. Kubota, K. Saito, M. Kimura, K. Ohara, and H. Motoda. Speeding up bipartite graph visualization method. In *Proceedings of the 24th Australasian Joint Conference on Artificial Intelligence*, pages 697–706. LNAI 7106, 2011.

[4] K. Kamada and S. Kawai. An algorithm for drawing general undirected graph. *Information Processing Letters*, 31:7–15, 1989.

[5] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 137–146, 2003.

[6] M. Kimura, K. Saito, and H. Motoda. Minimizing the spread of contamination by blocking links in a network. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*, pages 1175–1180, 2008.

[7] M. Kimura, K. Saito, and H. Motoda. Blocking links to minimize contamination spread in a social network. *ACM Transactions on Knowledge Discovery from Data*, 3:9:1–9:23, 2009.

[8] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46:604–632, 1999.

[9] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *Proceedings of the 2004 European Conference on Machine Learning (ECML'04)*, pages 217–226, 2004.

[10] A. N. Langville and C. D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1:335–380, 2005.

[11] A. Naud, S. Usui, N. Ueda, and T. Taniguchi. Visualization of documents and concepts in neuroinformatics with the 3d-se viewer. *Frontiers in Neuroinformatics*, 1:Article 7, 2007.

[12] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

[13] K. Saito, M. Kimura, K. Ohara, and H. Motoda. Learning continuous-time information diffusion model for social behavioral data analysis. In *Proceedings of the 1st Asian Conference on Machine Learning (ACML2009)*, pages 322–337. LNAI 5828, 2009.

[14] K. Saito, M. Kimura, K. Ohara, and H. Motoda. Selecting information diffusion models over social networks for behavioral analysis. In *Proceedings of the 2010 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2010)*, pages 180–195. LNAI 6323, 2010.

[15] W. Torgerson. *Theory and methods of scaling*. Wiley, New York, 1958.

[16] T. Yamada, K. Saito, and N. Ueda. Cross-entropy directed embedding of network data. In *Proceedings of the 20th International Conference on Machine Learning (ICML2003)*, pages 832–839, 2003.