Tracking Network Evolution and Their Applications in Structural Network Analysis

Tsunghan Wu, Cheng-Shang Chang, *Fellow, IEEE,* and Wanjiun Liao, *Fellow, IEEE*

Abstract—Structural network analysis, including node ranking, community detection, and link prediction, has received a lot of attention lately. In the literature, most works focused on the structural analysis of a single network. In this paper, we are particularly interested in how the network structure evolves over time. For this, we propose a general framework to track, model, and predict the dynamic network structures. Unlike some recent works that directly tracks the *adjacency* matrices of the networks, our framework utilizes the spectral graph theory to track the latent feature vectors obtained by a low-rank eigendecomposition of the *Laplacian* matrices of the networks. We then use the Finite Impulse Response (FIR) filter to model the evolution of the latent feature vector of each node. By solving a ridge regression problem, the parameters of the FIR filter can be learned and used for predicting the future network structures, including node ranking, community detection, and link prediction. To test the effectiveness of our framework, we perform various experiments based on our synthetic datasets and three real-world datasets. Our experimental results show that our framework is very effective in tracking latent feature vectors and predicting future network structures.

Index Terms—Network evolution, link prediction, community detection

1 Introduction

As online social networks become very popular in people's social life, structural analysis of networks has received a lot of attention lately. There are several important research topics in structural analysis of networks, including (i) centrality measures and node ranking [1], [2], [3], (ii) community detection and clustering [4], [5], [6], and (iii) similarity measures and link prediction [7]. In the literature, most works on structural network analysis are based on a single snapshot of a network. In this paper, we are particularly interested in how the network structure evolves over time by observing a sequence of networks indexed in time. Specifically, we ask the following two questions:

(i) Given a sequence of time-varying networks $\{G(t), t \geq 0\}$, what is the essential information (in a latent space) that we need to track for network structural analysis such as community detection, link prediction, and node ranking?
(ii) Given a sequence of time-varying networks $\{G(t), t = 0, 1, \ldots, T-1\}$, how can we predict the network structure at time T?

For the second question, most of the works in the literature focused on *one* specific objective of network structural analysis, e.g., link prediction [8], [9], [10], community analysis [11], [12], [13], evolutionary clustering [14], [15], or community evolution [16], [17], [18], [19], [20]. One common approach is to track some measures of the networks over time and use these measures to discover various properties of evolving networks [21], [22], [23]. Another approach is to model the dynamic evolutionary patterns from tracking network measures [24], [25]. All these approaches lack a *general* model to address the second question.

On the other hand, for the first question, there are many works on dynamic network analysis [10], [26], [27], [28], [29], [30]. Among these works, the latent space approaches in [10], [27] are of particular interest to us. Sarkar and Moore [27] extended the latent space approach in [31] for a static link prediction model to a dynamic network model. There they used a Hidden Markov Model (HMM) for modeling network evolution. Under the Markovian assumption, one can then use the matrix factorization approach to find the latent vectors for each time t. Instead of using the Markov assumption, a joint matrix factorization problem is proposed in [10] to find the latent vectors jointly for all t (Problem 1 in [10]). For such a problem, they proposed the block-coordinate gradient descent (BCGD) algorithm to obtain a local optimum of the problem. Both works [10], [27] used the *adjacency* matrices of networks to obtain the latent vectors via matrix factorization. Using the *adjacency* matrices might be reasonable for the link prediction problem targeted in these two papers. But it lacks theoretical supports for detecting the structure of a network.

Motivated by the success of the spectral graph theory (see e.g., [32], [33]) in community detection and clustering, in this paper we propose using the Laplacian matrices of networks to obtain the latent feature vectors. In comparison with the adjacency matrix, there are several advantages of the Laplacian matrix:

(i) The Laplacian matrix of a network is positive semidefinite and has an exact matrix factorization [32]. As such, the error of a low-rank matrix factor-

- *T. Wu is with the Graduate Institute of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. E-mail: d99921033@ntu.edu.tw*
- *C.-S. Chang is with the Institute of Communications Engineering, National Tsing Hua University, Hsinchu 300, Taiwan, R.O.C. E-mail: cschang@ee.nthu.edu.tw*
- *W. Liao is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. E-mail: wjliao@ntu.edu.tw*

Manuscript received Month Day, Year; revised Month Day, Year.

ization can be theoretically bounded by a function of its eigenvalues.

(ii) The number of components in a network is exactly the same as the number of zero eigenvalues of its Laplacian matrix. As such, the number of eigenvalues that are close to 0 is related to the community structure of a network. In particular, the second smallest eigenvector of the Laplacian matrix can be used for finding a good graph cut [33].

(iii) The resistance distance obtained from the pseudo-inverse of the Laplacian matrix [32] is a more effective dissimilarity measure than the geodesic distance obtained from the adjacency matrix (as the geodesic distance between two nodes in a network is only a positive integer bounded by the network diameter).

For a sequence of networks $\{G(t), t \geq 0\}$ with n (common) nodes, our specific steps to address these two problems are as follows:

(S1) For each time t, use a low-rank eigendecomposition (with rank K) for the Laplacian matrix of the network at time t to learn a $1 \times K$ feature vector $\mathbf{Z}_i(t)$ for each node i at time t.

(S2) Use the Finite Impulse Response (FIR) filter to track/learn the dynamics of the feature vectors $\{\mathbf{Z}_i(t), t = 0, 1, \ldots, T-1\}$.

(S3) Use the learned dynamics to predict the feature vectors of all the n nodes, i.e., $\mathbf{Z}_i(T)$, $i = 1, 2, \ldots, n$.

(S4) Use the predicted feature vectors for various applications of network analysis (e.g., link prediction, community prediction, and node ranking).

To the best of our knowledge, our work is the first one to propose a general framework that uses the Laplacian matrices of observed networks for the structural analysis of dynamic networks, including link prediction, community prediction, and node ranking. To test the effectiveness of our framework, we conduct various experiments on synthetic datasets and three real-world datasets, including Haggle, Infectious and Reality Mining obtained from the Koblenz Network Collection [34]. The synthetic datasets are generated by using the well-known stochastic block model with ground-truth communities and migrating users. Our experimental results for the synthetic datasets show that the feature vectors of migrating users can be tracked by an FIR filter with a proper order and these feature vectors also reveal the communities of the migrating users. For the link prediction problem, we compute the cosine similarity of the predicted latent vectors of two nodes. In comparison with the common-neighbor (CN) approach in the literature [35], our top-k predictor significantly outperforms that from the CN approach for migrating users. For the three real-world datasets, there are no ground-truth communities. As such, we compare our predicted community structure at time T with that from the spectral clustering algorithm [36], [37], [38] with a known adjacency matrix at time T. Our experimental results also show that the predicted community structure matches extremely well with that from the spectral clustering algorithm with a known adjacency matrix. For the link prediction problem, our top-k predictor still outperforms that from the CN approach for these three datasets. For the node ranking problem, our method is very effective in predicting/tracking important nodes in terms of the (refined) closeness centrality ([33], eq. (7.30)). In particular, for the Infectious dataset, the precision is roughly 80% for the top-15 predictor and over 60% for the top-5 predictor for most of the time.

The rest of this paper is organized as follows: In Section 2, we use the spectral theory and the FIR filter to track and predict the network evolution. We then discuss how our framework can be used for various structural analyses, including tracking community evolution, link prediction, and node ranking in Section 3. To test the effectiveness of our framework, we conduct various experiments in Section 4. Finally, we conclude the paper in Section 5, where we also discuss possible future research topics.

2 Tracking Network Evolution

2.1 Review of the Spectral Graph Theory

In this section, we briefly review the spectral graph theory and introduce the notations that we will use in this paper. For more details of the spectral graph theory, we refer to the books [32], [33]. Consider an undirected graph $G = (V, E)$ with the set of nodes V and the set of edges E. Let $n = |V|$ be the total number of nodes, $m = |E|$ be the total number of edges, and w_i, $i = 1, 2, \ldots, n$, be the degree of node i. Also, let A be the $n \times n$ adjacency matrix of the graph G, where its $(i, j)^{th}$ element is 1 if (i, j) is an edge in E and 0 otherwise. The (graph) Laplacian of G, denoted by L, is defined as $D - A$, where $D = diag(w_1, w_2, \ldots, w_n)$ is the $n \times n$ diagonal matrix with the diagonal elements being the degrees of the n nodes. It is well-known that the Laplacian L is symmetric and positive semi-definite. As such, all its n eigenvalues, $\lambda_i, i = 1, 2, \ldots, n$, are nonnegative. In this paper, we order the n eigenvalues such that

$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n.$$

Moreover, there exists an orthonormal basis of $n \times 1$ column vectors $\{\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_n\}$ such that $\mathbf{Z}_i$ is the eigenvector of L corresponding to the eigenvalue λ_i, i.e.,

$$L\mathbf{Z}_i = \lambda_i \mathbf{Z}_i.$$

As such, the Laplacian has the following eigendecomposition:

$$L = \sum_{i=1}^{n} \lambda_i \mathbf{Z}_i \mathbf{Z}_i^T, \tag{1}$$

where $\mathbf{Z}_i^T$ is the transpose of $\mathbf{Z}_i$.

It is well-known (see e.g., [33]) that the smallest eigenvalue, λ_1, is 0, and $\mathbf{Z}_1$ is the $n \times 1$ column vector with all its elements being $1/\sqrt{n}$. If, furthermore, the graph G is connected, then the second smallest eigenvalue, λ_2, known as the *algebraic connectivity*, is always positive. Thus, for a connected graph, one can define the pseudo-inverse of the Laplacian L as follows:

$$\Gamma = \sum_{i=2}^{n} \frac{1}{\lambda_i} \mathbf{Z}_i \mathbf{Z}_i^T. \tag{2}$$

The resistance distance between two nodes u and v, denoted by $d_{u,v}$, is then defined as

$$d_{u,v} = (\Gamma)_{u,u} + (\Gamma)_{v,v} - 2(\Gamma)_{u,v}, \tag{3}$$

where $(\Gamma)_{u,v}$ is the $(u,v)^{th}$ element of the matrix Γ. Using (2) yields

$$d_{u,v} = \sum_{i=2}^{n} \frac{(z_{i,u} - z_{i,v})^2}{\lambda_i}, \tag{4}$$

where $z_{i,u}$ is the u^{th} element of the vector $\mathbf{Z}_i$. The resistance distance is known to be a metric that satisfies the triangular inequality. Let

$$\epsilon_K = 2 \sum_{i=K+1}^{n} \frac{1}{\lambda_i}.$$

Since $||\mathbf{Z}_i|| = 1$, we have $(z_{i,u} - z_{i,v})^2 \leq 2$. Then it is easy to see from (4) that one can approximate the resistance distance $d_{u,v}$ by

$$\hat{d}_{u,v} = \sum_{i=2}^{K} \frac{(z_{i,u} - z_{i,v})^2}{\lambda_i} \tag{5}$$

such that

$$\hat{d}_{u,v} \leq d_{u,v} \leq \hat{d}_{u,v} + \epsilon_K. \tag{6}$$

The approximation error in (6) holds for any pair of two nodes. This suggests that one can approximate the resistance distance within the error bound ϵ_K by using the first K smallest eigenvalues and their corresponding eigenvectors of the Laplacian.

2.2 Modeling the System Dynamics

In this section, we propose our approach for tracking the evolution of a sequence of time-varying networks $\{G(t), 0 \leq t \leq T-1\}$. In view of the spectral theory discussed in the previous section, one can recover the Laplacian of a graph by using the eigendecomposition in (1). Thus, if we can track all the eigenvectors and the eigenvalues of a sequence of networks $\{G(t), 0 \leq t \leq T-1\}$, then we can recover the whole sequence of networks. However, tracking all the eigenvectors and the eigenvalues is very costly. For certain applications, such as community detection, link prediction, and node ranking, what we need is a distance metric between any pair of two nodes. As discussed in the previous section, such a distance metric can be well approximated by looking at the first K smallest eigenvalues and their corresponding eigenvectors of the Laplacian. Rooted in this, our approach for tracking the evolution of a sequence of time-varying networks $\{G(t), 0 \leq t \leq T-1\}$ is to track the first K smallest eigenvalues and the corresponding eigenvectors of the Laplacian matrices of these graphs. For this, we let $\lambda_i(t)$ be the i^{th} smallest eigenvalue of the Laplacian of the graph $G(t)$ and $\mathbf{Z}_i(t)$ be the corresponding eigenvector. Also, let $z_{i,k}(t)$ be the k^{th} element of $\mathbf{Z}_i(t)$. Ideally, if the sequence of networks evolves slowly, then one might expect that the i^{th} smallest eigenvalue and its corresponding eigenvector also evolve slowly and thus the order of the first K smallest eigenvalues/eigenvectors remains unchanged with respect to time. This is the assumption that we will use in this paper. Even under such an assumption, there is still a subtle problem that needs to be resolved. In the process of solving the eigendecomposition for the Laplacian of the graph $G(t)$, both $\mathbf{Z}_i(t)$ and $-\mathbf{Z}_i(t)$ are eligible for being the corresponding eigenvector of the eigenvalue $\lambda_i(t)$. For the purpose of tracking network evolution, we first compute the correlation between $\mathbf{Z}_i(t+1)$ and $\mathbf{Z}_i(t)$. If the correlation is nonnegative, then we select $\mathbf{Z}_i(t+1)$. On the other hand, if the correlation is negative, then we select $-\mathbf{Z}_i(t+1)$. We note that the sign of the correlation between $\mathbf{Z}_i(t+1)$ and $\mathbf{Z}_i(t)$ is the same as the sign of the inner product of these two vectors, i.e., $\mathbf{Z}_i(t+1)^T\mathbf{Z}_i(t)$. It is easy to see that this is equivalent to choosing

$$Z_i(t+1) = \text{argmin}_{Z \in \{Z_i(t+1), -Z_i(t+1)\}} ||Z - Z_i(t)||.$$

To track the network evolution, we need to identify a system model for the dynamics of the time series of these eigenvalues and eigenvectors. Such a problem is known as the *system identification* problem in the literature [39]. One of the most commonly used models for system identification problems is the linear system model. As such, we consider the Finite Impulse Response (FIR) filter (one of the simplest linear system models) for tracking each eigenvalue and each element in the eigenvectors. Specifically, we form the estimate $\{\hat{z}_{i,k}(t), t = 0, 1, \ldots, T-1\}$ by finding the impulse response $\alpha_{i,k,j}$, $j = 1, 2, \ldots, J$ (for some J) and the bias $b_{i,k}$ such that for $t = J, \ldots, T-1$,

$$\begin{aligned}\hat{z}_{i,k}(t) = \alpha_{i,k,1} z_{i,k}(t-1) + \alpha_{i,k,2} z_{i,k}(t-2) + \ldots \\ + \alpha_{i,k,J} z_{i,k}(t-J) + b_{i,k}.\end{aligned} \tag{7}$$

The best estimate can be solved by the following ridge regression.

$$\min_{\alpha_{i,k,j}, b_{i,k}} \sum_{t=J}^{T-1} \left(z_{i,k}(t) - \hat{z}_{i,k}(t)\right)^2 + \gamma \sum_{j=1}^{J} \|\alpha_{i,k,j}\|_2{}^2, \tag{8}$$

where γ is the regularization parameter. Similarly, for tracking the K eigenvalues, we form the estimate $\{\hat{\lambda}_i(t), t = 0, 1, \ldots, T-1\}$ by finding the impulse response $\beta_{i,j}$, $j = 1, 2, \ldots, J$ and the bias c_i such that for $t = J, \ldots, T-1$,

$$\begin{aligned}\hat{\lambda}_i(t) = \beta_{i,1} \lambda_i(t-1) + \beta_{i,2} \lambda_i(t-2) + \ldots \\ + \beta_{i,J} \lambda_i(t-J) + c_i.\end{aligned} \tag{9}$$

Once again, the best estimate can be solved by the following ridge regression.

$$\min_{\beta_{i,j}, c_i} \sum_{t=J}^{T-1} \left(\lambda_i(t) - \hat{\lambda}_i(t)\right)^2 + \gamma_2 \sum_{j=1}^{J} \|\beta_{i,j}\|_2{}^2, \tag{10}$$

where γ_2 is the regularization parameter.

We note that one can choose a much more complicated model than the FIR model proposed here. For instance, we solve the ridge regression problem in (7) *separately* for each dimension. One possible generalization is to solve the ridge regression problem *jointly* in the matrix form. But this not only increases the number of variables but also increases the computational cost. Also, according to several experimental results of ours, it seems that the evolution of the latent feature in one dimension is uncorrelated to that in another dimension. As such, using a matrix model for the regression problem is not recommended as it does not gain too much regarding the accuracy of the predicted feature

vectors. We also note that the first-order Markov assumption used in [10] might not be good enough for modeling social networks. From various experiments of ours, it seems that human behaviors are not just affected by their latest actions, and better predictive results can be obtained by increasing the order of the FIR filter to be more than 1. On the other hand, human behaviors are more likely to be influenced by recent events than old (obsolete) events, and the influence of an old event is generally discounted very quickly over time. As such, we believe that using the FIR filter should be enough to model the evolution of online social networks, and there is no need to use more complicated IIR filters.

We also note that the order of the eigenvectors might be changed *occasionally* in practice. When this happens, there is a transient period before our approach can regain its tracking capability.

2.3 Prediction of Network Evolution

Once we solve the ridge regressions in (8) and (10), we have all the needed parameters for modeling the system dynamics. As such, we can then use these for predicting the eigenvectors/eigenvalues of the network at time T. Specifically, we form the following estimates

$$\hat{z}_{i,k}(T) = \alpha_{i,k,1} z_{i,k}(T-1) + \alpha_{i,k,2} z_{i,k}(T-2) + \ldots + \alpha_{i,k,J} z_{i,k}(T-J) + b_{i,k}, \quad (11)$$

and

$$\hat{\lambda}_i(T) = \beta_{i,1}\lambda_i(T-1) + \beta_{i,2}\lambda_i(T-2) + \ldots + \beta_{i,J}\lambda_i(T-J) + c_i. \quad (12)$$

In view of (5), we can form the estimate of the resistance distance between two nodes u and v at time T by using

$$\hat{d}_{u,v}(T) = \sum_{i=2}^{K} \frac{(\hat{z}_{i,u}(T) - \hat{z}_{i,v}(T))^2}{\hat{\lambda}_i(T)}. \quad (13)$$

The details of our proposed approach for tracking/predicting time-varying graphs is shown in Algorithm 1.

2.4 Computational Complexity

Now we discuss the computational complexity of Algorithm 1. The time complexity for computing the degree of each node in Step (0) is $O(n^2)$ and that for the eigendecomposition in Step (1) is $O(n^3)$ [40], where n is the number of nodes in each network. In Step (2), the complexity of the dot product for the K eigenvectors is $O(nK)$. In general, K is much smaller than n. Since there are T networks, the computational complexity for step (0), (1) and (2) is $O(Tn^3)$. The time complexity for ridge regression and computing the resistance distances for all node pairs is $O([(T-J)J^2 + J^3]nK)$ and $O(Jn + n^2K)$, respectively. As J is much smaller than T, the overall computational complexity is $O(Tn\max[n^2, J^2K])$.

ALGORITHM 1: The Tracking Algorithm for Time-Varying Graphs

Input: A sequence of networks in time $\{G(t), 0 \le t \le T-1\}$ with the corresponding adjacency matrices $\{A(t), 0 \le t \le T-1\}$.

(0) Let $D(t) = diag(w_1(t), w_2(t), \ldots w_n(t))$, where $w_i(t) = \sum_{j=1}^{n} A_{i,j}(t)$ is the degree of node i in $G(t)$.

(1) Let $\lambda_1(t), \lambda_2(t), \ldots, \lambda_K(t)$ and $\mathbf{Z}_1(t), \mathbf{Z}_2(t), \ldots, \mathbf{Z}_K(t)$ be the K smallest eigenvalues and their corresponding eigenvectors of the Laplacian matrix $D(t) - A(t)$ (chosen to be orthogonal to each other).

(2) If $\mathbf{Z}_i(t)^T \mathbf{Z}_i(t-1) < 0$**, for** $t = 1, 2, \ldots, T-1$**, then** $\mathbf{Z}_i(t) = -\mathbf{Z}_i(t)$.

(3) Solve the ridge regressions in (8) and (10) to obtain the parameters for the FIR model for tracking eigenvectors/eigenvalues in (7) and (9).

(4) Use the estimates in (11) and (12) to predict the eigenvectors/eigenvalues of the network at time T. Approximate the resistance distance between two nodes u and v at time T by using (13).

2.5 Network Embedding and Latent Feature Vectors

In this section, we provide further insights of our tracking algorithm in Algorithm 1 by using *network embedding*. For this, let us define the latent feature vector of node u at time t, denoted by $\psi_u(t)$, as the $1 \times (K-1)$ row vector with its k^{th} element being $z_{k+1,u}(t)/\sqrt{\lambda_{k+1}(t)}$, i.e.,

$$\psi_u(t) = \left(\frac{z_{2,u}(t)}{\sqrt{\lambda_2(t)}}, \frac{z_{3,u}(t)}{\sqrt{\lambda_3(t)}}, \ldots, \frac{z_{K,u}(t)}{\sqrt{\lambda_K(t)}}\right). \quad (14)$$

In view of the latent feature vector in (14), we have created a mapping from every node in $G(t)$ to a vector in $\mathcal{R}^{K-1}$. Such a mapping is commonly known as *network embedding* in the literature. Now tracking the evolution of a sequence of networks in time is reduced to the problem of tracking a time series of finite dimensional vectors in a Euclidean space. Moreover, as each latent feature vector can be used for representing a node in a network, one can use that for various network analyses, such as community detection, link prediction, and node ranking (centrality). We will discuss these applications in the next section.

3 Applications

In this section, we discuss three possible applications of our tracking algorithm for time-varying networks: (i) tracking community evolution, (ii) link prediction, and (iii) node ranking.

3.1 Tracking Community Evolution

Detecting community structure in large complex networks has been a very hot research topic since the first paper appeared in the physics literature by Girvan and Newman [41]. The problem of detecting community structure in large complex networks is also known as the graph partitioning problem that divides a graph into several disjoint subgraphs, called clusters, blocks, or communities. As pointed

out in the review papers in [42] and [43], many algorithms have been developed by researchers from various research communities, including physicists, biologists, and computer scientists (see e.g., [44], [45], [46], [47], [48], [49] and references therein). However, it seems that a systematic method of tracking the evolution of communities is still lacking.

Our idea for tracking community evolution is rooted on the spectral clustering algorithm (see e.g., [50] for a tutorial). It is well-known (see e.g., the book [33]) that the number of zero eigenvalues of the Laplacian of a graph represents the number of components in that graph. As such, if there is a significant spectral gap between the K^{th} smallest eigenvalue and the $(K+1)^{th}$ smallest eigenvalue of the Laplacian of a graph, then that graph may be cut into K *communities*. To detect these K communities, the spectral clustering algorithm first embeds each node in a graph to a *latent feature vector* and then applies the K-means algorithm (see e.g., [51]) for clustering the n nodes into K communities based on their latent feature vectors. Certainly, one can use the latent feature vector in (14). However, for the purpose of community detection, we consider the following (simplified) latent feature vector

$$P_u(t) = (z_{1,u}(t), z_{2,u}(t), \ldots, z_{K,u}(t)). \tag{15}$$

The differences between the latent feature vector in (14) and that in (15) are the normalizing weights from the eigenvalues. As such, the K-means algorithm needed for the spectral clustering algorithm is replaced by the weighted K-means algorithm. We believe such a replacement is not crucial for the purpose of community detection if the networks are connected all the time. However, if for some time t, a network is separated into two or more components, then the latent feature vector in (14) cannot be used as $\lambda_2(t) = 0$. Thus, for the purpose of community detection, we believe the latent feature vector in (15) is a better choice.

Now with the latent feature vectors in (15), we can then apply the tracking algorithm in Algorithm 1 to track the evolution of communities. The detailed step is outlined in Algorithm 2. Note that for $t = T$, one can use the predicted eigenvectors in (11) of the network at time T for clustering the n nodes. Also, as the K-means algorithm is known to be a linear lime algorithm, the computational complexity of Algorithm 2 is also dominated by the eigendecomposition as in Algorithm 1.

We note that for the K-means algorithm, K centroids have to be chosen to represent K clusters at the beginning. If the sequence of networks evolves slowly, these feature vectors $P_i(t)$'s also vary slowly. As such, the centroids of the K communities also vary slowly and one can then use the centroids at time $t-1$ as the initial centroids at time t. However, for the choice of the initial K centroids at time 0, we simply choose them randomly. There are several papers that addressed the issue of choosing the initial centroids to improve the performance of the K-means algorithm (see e.g., [52], [53], [54], [55]).

3.2 Link Prediction

Liben-Nowell and Kleinberg [56] considered the following problem:

ALGORITHM 2: The Spectral Clustering Algorithm for Time-Varying Graphs

Input: A sequence of networks in time $\{G(t), 0 \le t \le T-1\}$ with the corresponding adjacency matrices $\{A(t), 0 \le t \le T-1\}$, and the number of communities K.
Output: A sequence of partitions of sets $\{S_1(t), S_2(t), \ldots, S_K(t)\}$.
(1) Use the tracking algorithm in Algorithm 1 to obtain $\mathbf{Z}_1(t), \mathbf{Z}_2(t), \ldots, \mathbf{Z}_K(t)$.
(2) Form the $n \times K$ matrix $\mathbf{Z}(t) = (z_{i,j}(t)) = [\mathbf{Z}_1(t), \mathbf{Z}_2(t), \ldots, \mathbf{Z}_K(t)]$ by stacking these K vectors in columns.
(3) Let $P_i(t)$ be the i^{th} row of $\mathbf{Z}(t)$. Treat $P_i(t)$ as a *feature vector* in $\mathcal{R}^K$ for node i at time t. Use the K-means algorithm to cluster these n feature vectors with the initial K centroids from the partition $\{S_1(t-1), S_2(t-1), \ldots, S_K(t-1)\}$.
(4) Assign node i to the set $S_k(t)$ if row i of $\mathbf{Z}(t)$ is assigned to the k^{th} cluster by the K-means algorithm.

Given a snapshot of a social network, can we infer which new interactions among its members are likely to occur in the near future?

Such a question was formulated as the *link prediction* problem and was addressed in [56] by considering various "proximity" measures. The link prediction problem is crucial to the success of constructing recommendation systems. There are various methods proposed in the literature (see e.g., the survey paper [57]), including collaborative filtering [58], spectral graph transformation [59], tensor factorization [8] and temporal matrix factorization [60]). Our approach for the link prediction problem is similar to that in [56]. One may use the approximation of the effective distance in (13) as the proximity measure. Given a sequence of networks $\{G(t), 0 \le t \le T-1\}$, one can then predict the k most likely links, called the top-k predictions, that are connected to node i at time T by selecting the k nodes that have the k smallest values of the approximated effective distance in (13).

As described in the previous section, there might be a problem for using the approximated effective distance in (13) when a network is separated into two or more components. When this happens, the approximated effective distance in (13) is not defined. To get around this problem, we propose using the latent feature vector in (15) and the cosine similarity measure as the proximity measure. Specifically, let

$$\hat{P}_u(T) = (\hat{z}_{1,u}(T), \hat{z}_{2,u}(T), \ldots, \hat{z}_{K,u}(T)), \tag{16}$$

where $\hat{z}_{i,k}(t)$ is the estimate in (11). For two nodes u and v at time T, the cosine similarity measure is defined as follows:

$$sim(u,v)^{COS} = \frac{\hat{P}_u^T(T) \cdot \hat{P}_v(T)}{\left\|\hat{P}_u(T)\right\| \cdot \left\|\hat{P}_v(T)\right\|}. \tag{17}$$

In addition to the cosine similarity measure, we note that one can also use other similarity measures in the literature

(see e.g., [33], [61]) for the two latent feature vectors, such as the Pearson's correlation coefficient.

3.3 Node Ranking

Analogous to the link prediction problem, one can also ask the following question:

Given a sequence of networks $\{G(t), 1 \leq t \leq T-1\}$, can we infer which nodes are important nodes at time T?

In order to define "important" nodes, there has to be a method for ranking the nodes in a graph. There are various notions of centralities that can be used for ranking nodes in the literature (see e.g., the book [33]), including degree centrality, eigenvector centrality, Katz centrality, PageRank [3], closeness centrality, and betweenness centrality. As we have already had the approximated effective distance in (13), we define the following (refined) closeness centrality ([33], eq. (7.30)) for node u at time T as follows:

$$C_u(T) = \frac{1}{n-1} \sum_{v \neq u} \frac{1}{\hat{d}_{u,v}(T)}, \tag{18}$$

where $\hat{d}_{u,v}(T)$ is the approximated effective distance defined in (13). Nodes with high closeness centrality in (18) are then considered to be important.

4 Experiments

4.1 Experiments on Synthetic Datasets

We first conduct our experiments on the synthetic datasets. The main reason for doing this is to test our method in a *controllable* environment so that we can gain insights of the effects of various parameters and thus better understand when our method could be effective.

4.1.1 Methods of Generating Synthetic Datasets

For $t = 0$, we first use the stochastic block model (SBM) to generate a network with n nodes and K blocks (communities or clusters). The stochastic block model is a generalization of the Erdös-Rényi random graph [62] and it has been widely used for generating random graphs that can be used for benchmarking community detection algorithms [63], [64]. In a stochastic block model with K blocks, the total number of nodes in the random graph is evenly distributed to these K blocks. The probability that there is an edge between two nodes within the same block is p_{in} and the probability that there is an edge between two nodes in two different blocks is p_{out}. These edges are generated independently. For our experiments, we set $n = 100$, $K = 4$, $p_{in} = 0.8$ and $p_{out} = 0.1$. Specifically, nodes $0, 1, \ldots, 24$ are in block 0, nodes $25, 26, \ldots, 49$ are in block 1, nodes $50, 51, \ldots, 74$ are in block 2, and nodes $75, 76, \ldots, 99$ are in block 3.

To model the needed network evolution, a subset of nodes, called *migrating nodes*, are selected to migrate from blocks to blocks. To ensure that the sequence of networks evolves slowly, only a small set of migrating nodes are selected. For our experiments, we choose the following set of migrating nodes:

$$\{1, 15, 29, 43, 57, 71, 85, 99\}.$$

TABLE 1
The Blocks for Migrating Nodes at $t = 0$ and $t = 99$ Generated by Method 1 and Method 2

Node	$t = 0$	$t = 99$ **(M1)**	$t = 99$ **(M2)**
1	0	3	3
15	0	3	2
29	1	0	0
43	1	0	1
57	2	1	0
71	2	1	0
85	3	2	1
99	3	2	1

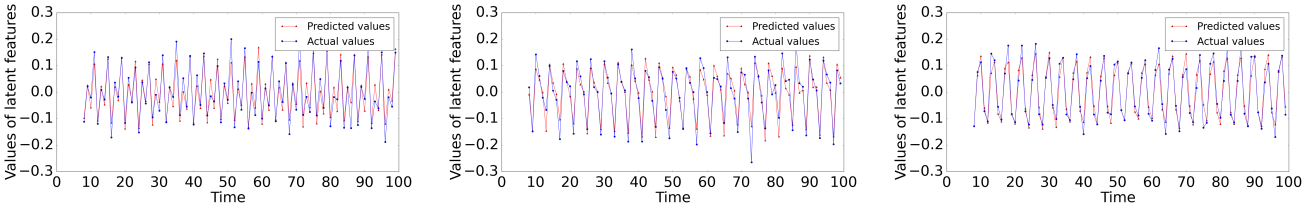
When a migrating node moves from one block at time t to another block at time $t+1$, we remove all its edges at time t and randomly add new edges to this migrating node at time $t+1$ according to the construction of the SBM, i.e., with probability p_{in} (resp. p_{out}) an edge is added between this migrating node and another node in the same block (resp. different blocks).

We consider two methods for modeling the (deterministic) migrating patterns of the migrating nodes:

(i) Method 1: a migrating node in block j at time t will migrate to block $((j+1) \bmod K)$ at time $t+1$.
(ii) Method 2: we increase additional complexity to the migrating pattern by allowing the migrating pattern to be dependent on the index of a migrating node and time. Specifically, the migrating node i in block j at time t will migrate to block $((j + ((t \bmod i) + 1)) \bmod K)$ at time $t+1$.

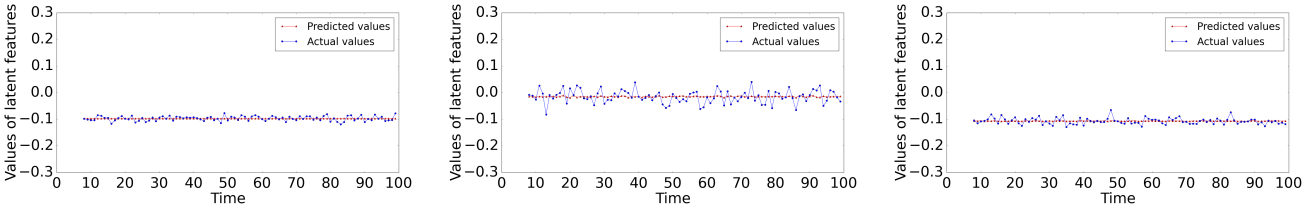
4.1.2 Tracking Latent Feature Vectors

In this section, we report our experimental results for using the FIR model to track the feature vectors. In our experiments, we set $T = 99$ and $J = 8$ for Algorithm 1. In Fig. 1 and Fig. 2, we compare the actual latent features (from the feature vectors for the network at time t) and the predicted latent features (predicted by (11)) in time. Since $K = 4$, each feature vector has four latent features. However, as discussed in Section 2.1, the values of the first latent features are fixed constants over time (i.e., $1/\sqrt{n}$ or $-1/\sqrt{n}$). Therefore, we only show the other three latent features in the following figures. In Fig. 1, we show the comparison results for node 1 (a migrating node). Since the migrating pattern designed for Method 1 is periodic with period 4, we expect to see that the features values are also periodic with the same period. As shown in this figure, the predicted values are very close to the actual values in our experimental results. It means our FIR model can track the evolutions of migrating nodes well. Additionally, in Fig. 2, we show the values of feature vectors of node 0 (a non-migrating node). As shown in this figure, the actual latent features of this non-migrating node do not fluctuate a lot and we also have steady predicted latent features from our model. It is a reasonable result because non-migrating nodes are expected to have static latent features over time. In Fig. 3 and Fig. 4, we show similar results for the synthetic networks generated by Method 2.



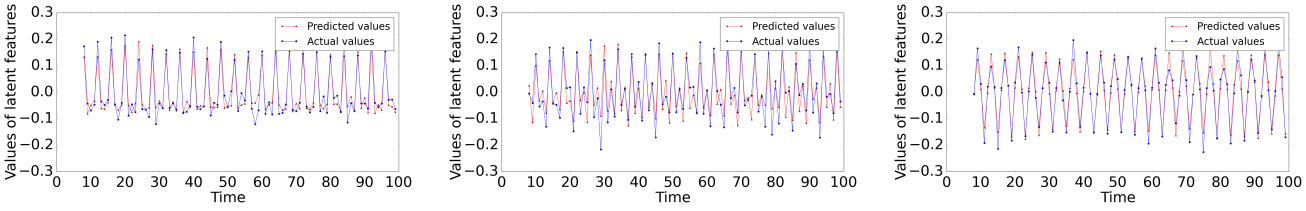
(a) The 2nd latent feature (b) The 3rd latent feature (c) The 4th latent feature

Fig. 1. Comparisons of actual latent feature vectors and predicted latent feature vectors over time of node 1 (a migrating node) for Method 1.



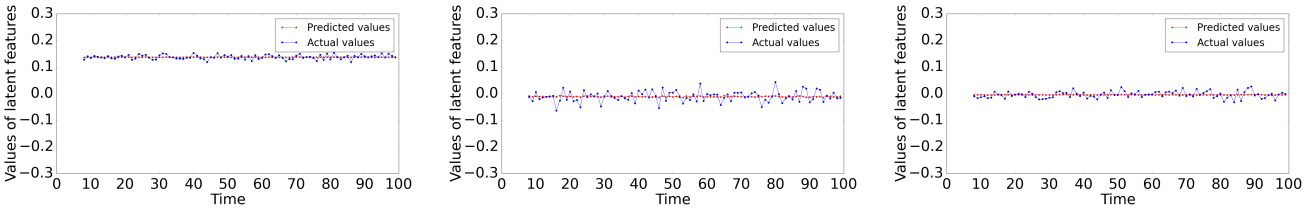
(a) The 2nd latent feature (b) The 3rd latent feature (c) The 4th latent feature

Fig. 2. Comparisons of actual latent feature vectors and predicted latent feature vectors over time of node 0 (a non-migrating node) for Method 1.



(a) The 2nd latent feature (b) The 3rd latent feature (c) The 4th latent feature

Fig. 3. Comparisons of actual latent feature vectors and predicted latent feature vectors over time of node 1 (a migrating node) for Method 2.



(a) The 2nd latent feature (b) The 3rd latent feature (c) The 4th latent feature

Fig. 4. Comparisons of actual latent feature vectors and predicted latent feature vectors over time of node 0 (a non-migrating node) for Method 2.

4.1.3 Prediction of Latent Feature Vectors

In the previous section, we have shown that the FIR model can be used for tracking latent feature vectors. In the section, we further report our experimental results for *predicting* latent feature vectors. For this, we use the networks for $0 \leq t \leq T-1$ as the training set and the network at time T as the testing set. Once again, T is set to be 99. Our objective is to predict the feature vectors for all nodes at time T by using (11).

As shown in Fig. 5, the predicted values and the actual values are very close to each other for the synthetic dataset generated by Method 1. It means that we can predict the feature vectors with a high degree of accuracy. Moreover, the predicted feature vectors of a node can be used for determining the block of that node (and we will use the spectral clustering algorithm in Algorithm 2 to verify this in the next section). In Fig. 5, there are four different colors representing the four different blocks: red for block 0, green for block 1, yellow for block 2 and blue for block 3. One can easily see from the four colors in Fig. 5(a) that the blocks for these n nodes at time 0 (as the first eigenvector remains unchanged over time). For example, nodes $0, 1, \ldots 24$ are in the red region which means they are in block 0 at time 0. However, the *migrating nodes* migrate to different blocks during $[0, T]$. One can observe that there are some *peaks* in Fig. 5(b), Fig. 5(c) and Fig. 5(d). Those peaks indicate that some of the migrating nodes no longer belong to their original blocks and they move to other blocks at time T. In Table 1, we show the ground-truth blocks for the *migrating users* at time 0 and time T. Furthermore, we number and color each *migrating user* with the same color of its block at time T. As shown in Fig. 5 and Fig. 6, the predicted values of latent features of each migrating node at time T match very well with those of the other nodes in the same block.

TABLE 2
The Community Prediction Results For Two Synthetic Datasets Generated by Method 1 and Method 2 Respectively

Method	Accuracy	NMI
Method 1	1.00	1.00
Method 2	0.99	0.97

As such, we also expect the spectral clustering algorithm in Algorithm 2 can be used for predicting the block of each migrating node. We take node 1 for example (and the other migrating nodes, node 15, node 29, node 43, node 57, node 71, node 85, and node 99, can be verified in the same way). First, we note from Table 1 that node 1 is in block 3 at time T. As such, we expect that the latent features of node 1 will be similar to the latent features of the other nodes in block 3. In Fig. 5, we mark node 1 by a blue circle. From Fig. 5, one can easily observe that node 1 indeed has similar latent features to the nodes in the same block. Our experimental results of these predicted values of feature vectors reveal that the spectral clustering algorithm in Algorithm 2 will work well. In Fig. 6, we observe the same results for the synthetic dataset generated by Method 2.

4.1.4 Community Prediction

In this section, we report the community prediction results for all the n nodes at time T by using the spectral clustering algorithm in Algorithm 2. We use two metrics "accuracy" and "NMI" (the Normalized Mutual Information measure) to evaluate our algorithm. The ground-truth blocks for the migrating nodes at $t = 99$ are shown in Table 1 for both datasets from Method 1 and Method 2, and the ground-truth blocks for the non-migrating nodes at $t = 99$ are the same as their ground-truth blocks at $t = 0$. Let n_c be the number of nodes that are correctly predicted by Algorithm 2 (after finding the permutation of the four output sets of Algorithm 2 that maximizes the number of correctly predicted nodes). Then the accuracy is defined as

$$Acc = \frac{n_c}{n}. \tag{19}$$

As shown in Table 2, for the synthetic dataset generated by Method 1, the accuracy is 100%, i.e., all the nodes are correctly predicted. For the synthetic dataset generated by Method 2, the accuracy is also near 100%. This shows that the spectral clustering algorithm in Algorithm 2 is very effective in tracking/predicting community evolution. As shown in Table 2, Algorithm 2 also yields very high NMI for both synthetic datasets.

4.1.5 Link Prediction

In this section, we report our experimental results for link prediction by using the same synthetic datasets as those in the previous section. For our link prediction experiments, we only report the results obtained by using the cosine similarity measure in (17). The experimental results obtained by using the Pearson's correlation coefficient are similar. A node pair with a higher cosine similarity score is considered to have a higher probability to form a link between them. We compare our method with the common-neighbors (CN) approach that is based on counting the number of common neighbors between each node pair. There are many methods proposed in the literature for the link prediction problem (see e.g., [65]), but there does not exist a single dominating approach for all the cases. The reason we choose the common-neighbors approach is that it is known to be an intuitive and very effective method (see e.g., [35]). In Table 3, we show the precision and recall of the top-k predictor for $k = 5, 10, 15$ and 20 for node 0 (a non-migrating node) and node 1 (a migrating node) at time $t = 99$ for the synthetic dataset by Method 1. The results for Method 2 are shown in Table 4. These results are obtained by averaging 100 experiments. From these two tables, one can see that our method is significantly better than the CN approach for node 1 (a migrating node). This is because a migrating node removes all its previous links and establishes new links according to its new community structure (from our synthetic methods). As the CN approach cannot predict the new community structure and thus fails to predict the new links of a migrating node. On the other hand, as our method can predict the new community structure of a migrating node (as illustrated in the previous section), our method can predict new links with a much higher precision. Even for a non-migrating node, such as node 0, our method still outperforms the CN approach.

4.2 Experiments on Real-world Datasets

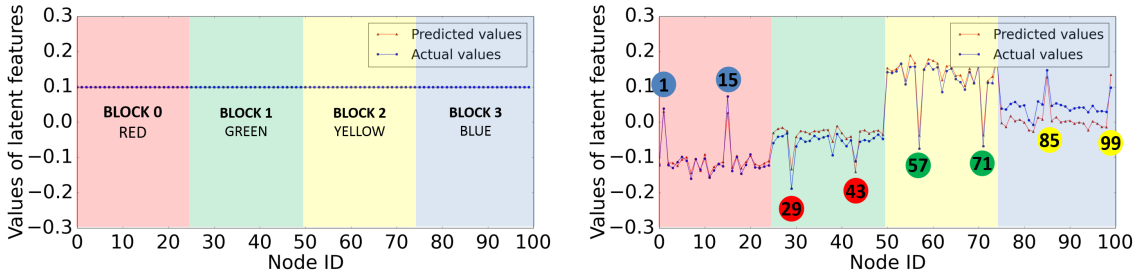
Besides the synthetic data, we also conduct experiments on three real-world datasets in this section. We evaluate our method for three applications of structural network analysis, community prediction, link prediction, and node ranking.

4.2.1 Datasets

The three real-world datasets are Haggle, Infectious and RM (Reality Mining) obtained from the Koblenz Network Collection [34].

(i) Haggle [66]: This network records human contacts by carried wireless devices in the University of California, San Diego (UCSD) during 77 days. A node represents a person, and a timestamped link between two nodes represents that two people contacted each other at a specific time.
(ii) Infectious [67]: The network records the face-to-face contacts between visitors during the exhibition INFECTIOUS: STAY AWAY in 2009 at the Science Gallery in Dublin. A node represents a visitor, and a link represents a contact between two visitors for at least 20 seconds.
(iii) Reality Mining (RM) [68]: The network records the human contacts between 100 students or faculty of the Massachusetts Institute of Technology (MIT) in 2004 during 9 months. A node represents a student, and a link represents a physical contact.

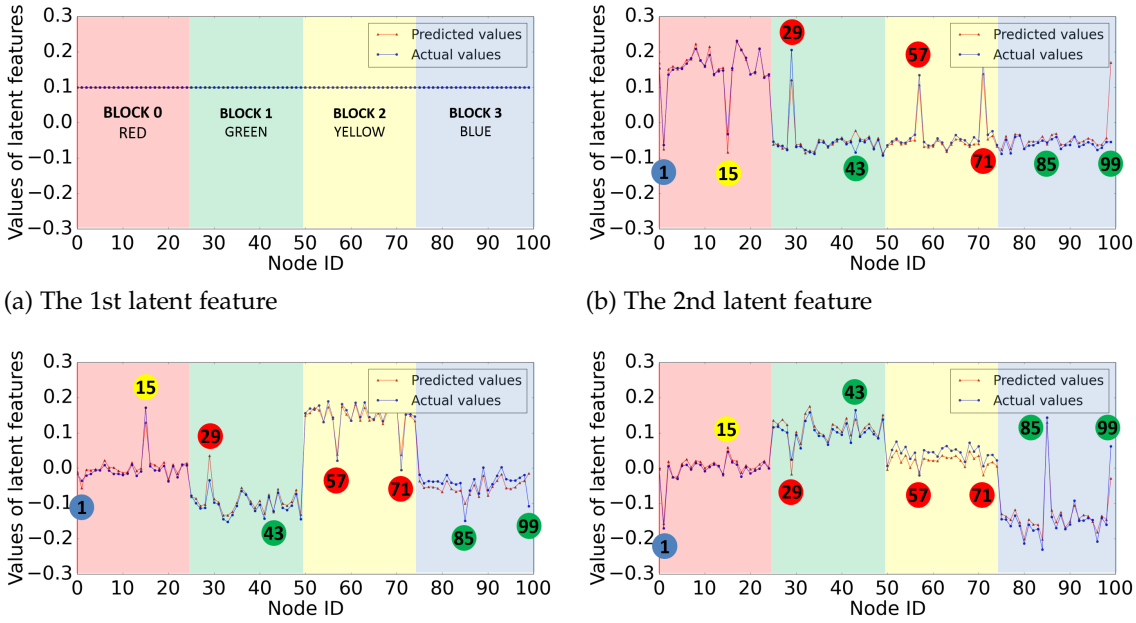
In order to obtain a sequence of discrete-time networks $\{G_p(t), t \geq 0\}$ for each dataset, we first partition the time duration of a dataset into *time slots* (with a fixed length), indexed from $t = 0, 1, 2, \ldots$. The links with timestamps in the t^{th} same slot are added to the network $G_p(t)$. To ensure the slow evolution of such a sequence of networks,



(a) The 1st latent feature (b) The 2nd latent feature

(c) The 3rd latent feature (d) The 4th latent feature

Fig. 5. Comparisons of actual latent feature vectors and predicted latent feature vectors at $t = 99$ for Method 1. Four different colors are used for representing the four blocks. The numbered nodes are migrating nodes, and their colors denote their blocks at $t = 99$.



(a) The 1st latent feature (b) The 2nd latent feature

(c) The 3rd latent feature (d) The 4th latent feature

Fig. 6. Comparisons of actual latent feature vectors and predicted latent feature vectors at $t = 99$ for Method 2. Four different colors are used for representing the four blocks. The numbered nodes are migrating nodes, and their colors denote their blocks at $t = 99$.

TABLE 3
Precision/Recall of The Top-k Predictors at Time $t = 99$ by Using The Cosine Similarity Measure (COS) in (17) and The Number of Common Neighbors (CN) for The Synthetic Dataset Generated by Method 1

Node 0	top-5	top-10	top-15	top-20	top-25	top-30
COS	0.928/0.156	0.904/0.306	0.922/0.470	0.914/0.618	0.870/0.733	0.767/0.779
CN	0.758/0.127	0.776/0.262	0.807/0.411	0.840/0.569	0.803/0.677	0.688/0.700
Node 1	top-5	top-10	top-15	top-20	top-25	top-30
COS	0.896/0.151	0.836/0.281	0.875/0.444	0.854/0.576	0.808/0.680	0.707/0.715
CN	0.128/0.021	0.124/0.041	0.133/0.066	0.135/0.090	0.146/0.121	0.184/0.184

TABLE 4
Precision/Recall of The Top-k Predictors at Time $t = 99$ by Using The Cosine Similarity Measure (COS) in (17) and The Number of Common Neighbors (CN) for The Synthetic Dataset Generated by Method 2

Node 0	top-5	top-10	top-15	top-20	top-25	top-30
COS	0.904/0.158	0.921/0.319	0.916/0.479	0.903/0.621	0.878/0.770	0.776/0.806
CN	0.744/0.129	0.810/0.281	0.811/0.424	0.828/0.569	0.808/0.710	0.695/0.722
Node 1	top-5	top-10	top-15	top-20	top-25	top-30
COS	0.816/0.141	0.781/0.270	0.846/0.441	0.847/0.581	0.800/0.700	0.672/0.693
CN	0.124/0.021	0.109/0.037	0.107/0.054	0.127/0.085	0.101/0.087	0.139/0.141

TABLE 5
Statistics and Parameters of The Three Datasets

Dataset	**# nodes**	**# links@T**	**# snapshots**	J	K
Haggle	60	941	150	10	30
Infectious	52	172	250	5	40
RM	43	236	200	15	30

TABLE 6
NMI Results for The Three Datasets

Dataset	**NMI**
Haggle	1.00
Infectious	1.00
RM	0.95

we adopt a smoothing mechanism, called the *sliding window mechanism*, that aggregates several consecutive time slots into a time window. Specifically, let T_w be the number of time slots in a time window. Then we construct another sequence of discrete-time networks $\{G(t), t \geq 0\}$ by adding all the links in $\{G_p(s),\ s = t, t+1, \ldots, t+T_w-1\}$ in $G(t)$. Each network $G(t)$ generated this way is called a *snapshot* of the dataset at time t.

In Table 5, we summarize the statistics of the three datasets, including the number of nodes and the number of links. It should be noted that the number of links in these three networks might vary with respect to time and we only show the number of links in each network at the last snapshot. The third column of Table 5 shows the total number of snapshots in these three datasets. The order of the FIR filter J and the number of eigenvectors K used in our tracking method for each dataset are also shown in the last two columns of Table 5. To visualize these three networks, we also plot the networks at the last snapshot in Fig. 7. As shown in Fig. 7, it seems that both the Haggle network and the RM network only have a densely connected component and show no obvious community structure. On the other hand, the Infectious network exhibits a much clearer community structure. All these three networks are in the category of human contact networks. Through our experiments, we will demonstrate that our proposed method is capable of identifying/capturing the structure of human interactions.

4.2.2 Community Prediction

In this section, we predict the community structure in the network at time T and use NMI as our evaluation metric. Traditionally, a community detection algorithm divides the network into several clusters based on the graph. But we *predict* the community structure of a graph without having the information of the whole graph. For our spectral clustering algorithm for time-varying graphs in Algorithm 2, we use the predicted latent feature vector of each node to construct the community structure. To show the effectiveness of our method, we first directly perform spectral clustering on the network at time T and use such a result as the ground-truth community structure at time T. We then compare that ground-truth community structure with the predicted community structure from our method. In Table 6, we show the NMI scores for these three datasets. The high NMI scores indicate that our method can not only track the network evolutions but also predict the community structures in the future.

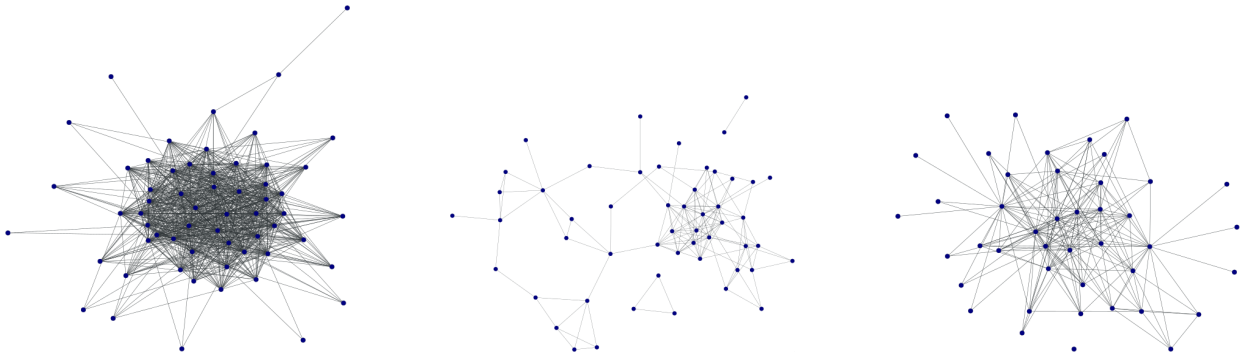
4.2.3 Link Prediction

In this section, we report our link prediction results for these three real-world datasets. As our experiments for the synthetic datasets, we also use the cosine similarity measure (COS) in (17) for link prediction and compare that with the common-neighbors (CN) approach. In Fig. 8, we plot the precision of the top-k predictor for each dataset (as a function of k). Since these three networks vary slowly, the structures of two consecutive networks are similar. As such, the common-neighbors approach can yield a high precision for link prediction. Nevertheless, we still can obtain better precisions than those of the CN approach (as shown in Fig. 8). The experimental results on the three datasets demonstrate that our model provides a useful mechanism for tracking the network evolutions of human interactions. We can predict the links that will appear in the future only based on the feature vector of each node instead of the whole topologies of networks.

4.2.4 Node Ranking

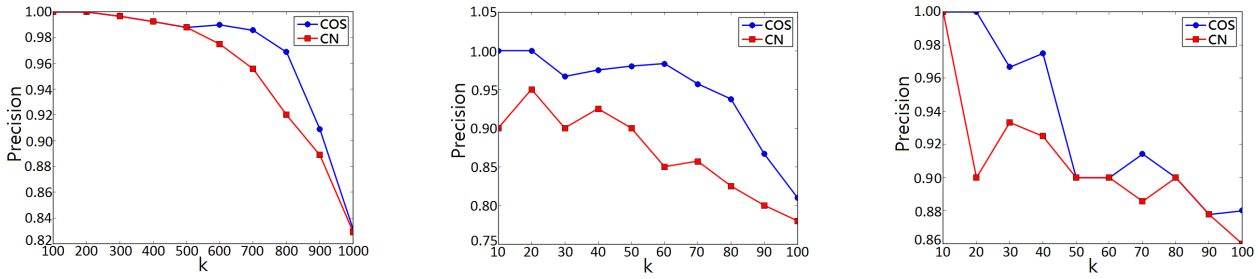
In this section, we test the effectiveness of our approach for node ranking by using the Infectious dataset. For the ground-truth centrality, we first compute the (refined) closeness centrality ([33], eq. (7.30)) by using the *geodesic distance* between each node pair in the original network at each time t. We then use the predicted latent feature vectors obtained in the previous section to compute the approximated effective distance as follows:

$$\hat{d}_{u,v}(t) = \sum_{i=2}^{K} (\hat{z}_{i,u}(t) - \hat{z}_{i,v}(t))^2, \tag{20}$$



(a) Haggle (b) Infectious (c) Reality Mining

Fig. 7. Visualization of the networks at time T. Haggle and RM have a densely connected component and no apparent community structure.



(a) Haggle (b) Infectious (c) Reality Mining

Fig. 8. Comparisons of the link prediction results of top-k links predicted by our method (COS) and common-neighbors (CN).

where $\hat{z}_{i,u}(t)$ is the predicted i^{th} latent feature of node u at time t. The approximated effective distance between each node pair is then used for computing the predicted closeness centrality as described in (18). In Fig. 9, we show the precision values of the top-5, top-10 and top-15 closeness ranking predictors over time. The precision of the top-k predictor is defined as the ratio of the number of predicted top-k nodes that are also actual top-k nodes to the number k. For the top-15 predictor, the precision is very good, roughly 80%. The precision values of the top-5 predictor, though not as good as those of the top-15 predictor, are over 60% for most of the time. In Fig. 10, we also plot the actual ranking and the predicted ranking for nodes 1, 19 and 52 over time. The rankings of these three nodes are within top-20 for most of the time. When a node's ranking is within top-20, its predicted ranking is quite close to its actual ranking. However, when a node's ranking is out of top-20, the predicted ranking could be quite different from the actual ranking as shown in Fig. 10. For node 1, its ranking becomes lower at the end time period (i.e., the time stamp is approximately 250), and the predicting error also becomes larger during that period. The same phenomenon can also be seen for node 52 during the beginning time period. Our experimental results show that our method is effective in predicting/tracking important nodes, but its performance for low-ranked nodes might not be good. Therefore, when a node's ranking becomes lower (and this node becomes less important), the performance of node ranking prediction for this node will also become worse. This might be due to

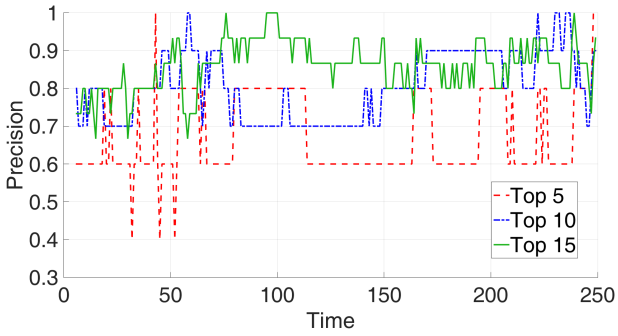
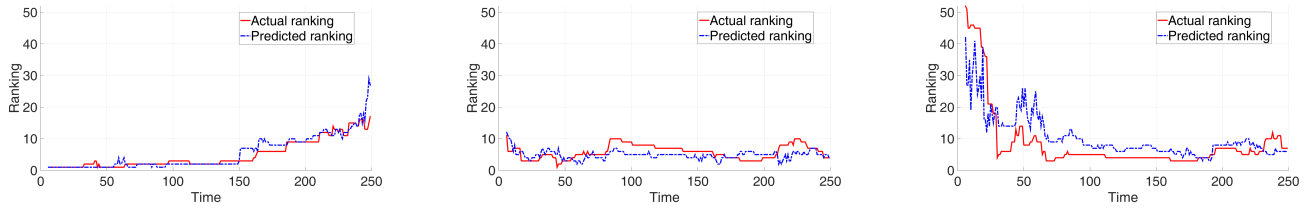


Fig. 9. The precision values of the top-5, top-10 and top-15 closeness ranking predictors over time.

the fact that we use the approximated effective distance that only captures the behavior of important nodes.

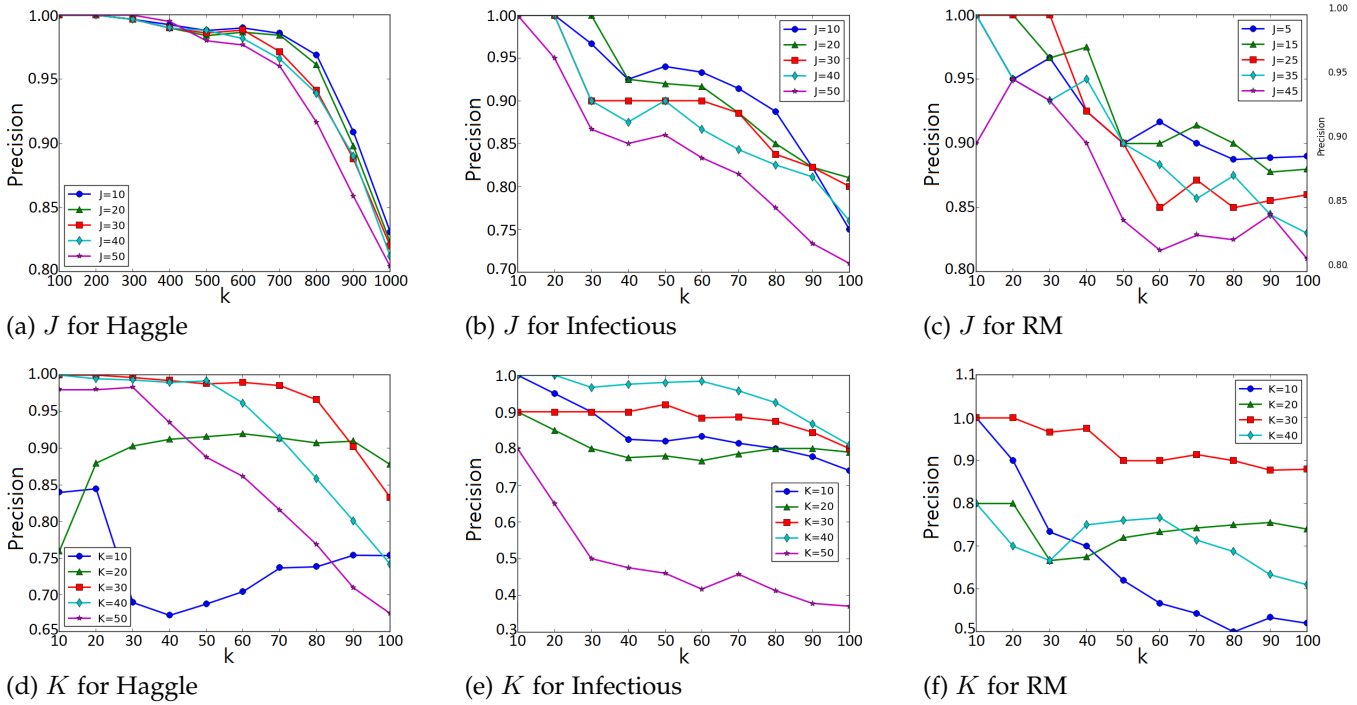
4.2.5 *Parameter Settings*

In our algorithm, there are two key parameters that need to be specified: the order of the FIR filter J and the number of eigenvectors K (i.e., the length of the latent feature vectors of each node). In Fig. 11, we compare several different parameter settings for each network by showing the top-k link prediction results. As shown in this figure, choosing the proper parameters does affect the performance. The two parameters J and K in the last two columns of Table 5 are used for our experiments conducted in Section 4.2.2 and Section 4.2.3. These values of J and K have not been fine-



(a) Node 1 (b) Node 19 (c) Node 52

Fig. 10. Comparisons of actual closeness rankings and predicted closeness rankings over time.



(a) J for Haggle (b) J for Infectious (c) J for RM

(d) K for Haggle (e) K for Infectious (f) K for RM

Fig. 11. Comparisons of different values of J and K for each network.

tuned as we simply compare different values and choose a relative better one for our settings.

In Fig. 11, we can observe that our model does not need a large J for the FIR filter. There are two insights for this observation. The first one is if we would like to predict human behaviors, we do not need to know too much information in the past. More recent historical data have larger impacts. The second is the cycles (periodic patterns) of human behaviors might be rather short. As such, we do not need a large J (i.e., many parameters) to model a single behavior (e.g., contact with each other) of human beings. In general, human behaviors are too complex to predict, but for certain periodic routines (as shown in the three datasets) they can be captured by using simple models like ours. Furthermore, choosing a large J might cause overfitting and that reduces the prediction accuracy.

The problem of selecting a proper K depends on the network structure. From the results, different networks need different values of K. Choosing a very small K might not have enough information to characterize the network structure. On the other hand, if we choose a large K, then we need to track a lot of eigenvectors and that might lead to large tracking errors. As shown in Fig. 11, using the largest K do not have the best performance.

5 Conclusion

In this paper, we proposed an effective framework to track, model, and predict the dynamic network structures. Instead of tracking the adjacency matrices of the networks, our framework tracks the evolution of network structure by tracking the latent feature vector of each node obtained from the eigendecomposition of the Laplacian matrices. To learn the dynamic of the networks, we applied the FIR filter to model the evolution of the latent feature vector of each node. Once the dynamic of the networks is learned, it can then be used for predicting the future network structures, including community detection, link prediction, and node ranking. Our experimental results for both the synthetic datasets and the three real-world datasets show that our framework is very effective in tracking latent feature vectors and predicting future network structures. In particular, our experimental results for the three real-world datasets also suggested that human behaviors are related to their recent actions and one can select a proper order J of the FIR filter to obtain good prediction results for different networks and different scenarios. As inspired by [69], [70], one possible

extension for our framework is to deploy the distributed filters to track the latent features for different users.

ACKNOWLEDGMENTS

This work was supported by Ministry of Science and Technology, Taiwan, R.O.C., under grant numbers MOST 104-2221-E-002-081-MY3 and 105-2221-E-007-037-MY3.

REFERENCES

[1] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.
[2] ——, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1979.
[3] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107–117, 1998.
[4] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, no. 6, p. 066133, 2004.
[5] R. Lambiotte, "Multi-scale modularity in complex networks," in *Proc. IEEE Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2010, pp. 546–553.
[6] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," *Proceedings of the National Academy of Sciences*, vol. 107, no. 29, pp. 12 755–12 760, 2010.
[7] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proc. ACM Int. Conf. Information and Knowledge Management*, 2003, pp. 556–559.
[8] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Trans. Knowledge Discovery from Data*, vol. 5, no. 2, 2011.
[9] P. Sarkar, D. Chakrabarti, and M. Jordan, "Nonparametric link prediction in dynamic networks," *arXiv preprint arXiv:1206.6394*, 2012.
[10] L. Zhu, D. Guo, J. Yin, G. Ver Steeg, and A. Galstyan, "Scalable temporal latent space inference for link prediction in dynamic social networks," *IEEE Trans. Knowledge and Data Engineering*, vol. 28, no. 10, pp. 2765–2777, 2016.
[11] C. C. Bilgin and B. Yener, "Dynamic network evolution: Models, clustering, anomaly detection," *IEEE Networks*, 2006.
[12] Y. Chi, S. Zhu, X. Song, J. Tatemura, and B. L. Tseng, "Structural and temporal analysis of the blogosphere through community factorization," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2007, pp. 163–172.
[13] N. P. Nguyen, T. N. Dinh, Y. Shen, and M. T. Thai, "Dynamic social community detection and its applications," *PLOS ONE*, vol. 9, no. 4, p. e91431, 2014.
[14] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2006, pp. 554–560.
[15] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "On evolutionary spectral clustering," *ACM Trans. Knowledge Discovery from Data*, vol. 3, no. 4, p. 17, 2009.
[16] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Proc. IEEE Int. Conf. Advances in Social Networks Analysis and Mining (ASONAM'10)*, 2010, pp. 176–183.
[17] P. Bródka, S. Saganowski, and P. Kazienko, "Ged: the method for group evolution discovery in social networks," *Social Network Analysis and Mining*, vol. 3, no. 1, pp. 1–14, 2013.
[18] M. Takaffoli, R. Rabbany, and O. R. Zaïane, "Community evolution prediction in dynamic social networks," in *Proc. IEEE Int. Conf. Advances in Social Networks Analysis and Mining (ASONAM'14)*, 2014, pp. 9–16.
[19] S. Saganowski, B. Gliwa, P. Bródka, A. Zygmunt, P. Kazienko, and J. Koźlak, "Predicting community evolution in social networks," *Entropy*, vol. 17, no. 5, pp. 3053–3096, 2015.
[20] N. İlhan and Ş. G. Öğüdücü, "Predicting community evolution based on time series modeling," in *Proc. IEEE/ACM Int. Conf. Advances in Social Networks Analysis and Mining (ASONAM'15)*, 2015, pp. 1509–1516.
[21] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery in Data Mining*, 2005, pp. 177–187.
[22] H. Tong, S. Papadimitriou, P. S. Yu, and C. Faloutsos, "Fast monitoring proximity and centrality on time-evolving bipartite graphs," *Statistical Analysis and Data Mining*, vol. 1, no. 3, pp. 142–156, 2008.
[23] F. Fogelman-Soulié, D. Perrotta, J. Piskorski, and S. Ralf, "Evolving networks," *Mining Massive Data Sets for Security: Advances in Data Mining, Search, Social Networks and Text Mining, and Their Applications to Security*, vol. 19, p. 198, 2008.
[24] B. Bringmann, M. Berlingerio, F. Bonchi, and A. Gionis, "Learning and predicting the evolution of social networks," *IEEE Intelligent Systems*, vol. 25, no. 4, pp. 26–35, 2010.
[25] R. Michalski, P. Kazienko, and D. Król, "Predicting social network measures using machine learning approach," in *Proc. IEEE Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM'12)*, 2012, pp. 1056–1059.
[26] K. M. Carley, *Dynamic network analysis*. Citeseer, 2003.
[27] P. Sarkar and A. W. Moore, "Dynamic social network analysis using latent space models," *SIGKDD Explor. Newsl.*, vol. 7, no. 2, pp. 31–40, Dec. 2005.
[28] T. Y. Berger-Wolf and J. Saia, "A framework for analysis of dynamic social networks," in *Proc. ACM SIGKDD Int. Conf.on Knowledge Discovery and Data Mining*, New York, NY, USA, 2006, pp. 523–528.
[29] M. Lahiri and T. Y. Berger-Wolf, "Mining periodic behavior in dynamic social networks," in *Proc. IEEE Int. Conf. on Data Mining*, 2008, pp. 373–382.
[30] C. Aggarwal and K. Subbian, "Evolutionary network analysis: A survey," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 10:1–10:36, May 2014.
[31] P. D. Hoff, A. E. Raftery, and M. S. Handcock, "Latent space approaches to social network analysis," *J. American Statistical Association*, vol. 97, no. 460, pp. 1090–1098, 2002.
[32] D. Spielman, "Spectral graph theory," *Lecture Notes, Yale University*, pp. 740–0776, 2009.
[33] M. Newman, *Networks: an introduction*. OUP Oxford, 2009.
[34] "konect network dataset – konect, october 2016," http://konect.uni-koblenz.de/networks/konect.
[35] L. Yao, L. Wang, L. Pan, and K. Yao, "Link prediction based on common-neighbors for dynamic social network," *Procedia Computer Science*, vol. 83, pp. 82–89, 2016.
[36] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
[37] S. X. Yu and J. Shi, "Multiclass spectral clustering," in *Proc. IEEE Int. Conf. Computer Vision*, 2003, pp. 313–319.
[38] I. Dhillon, Y. Guan, and B. Kulis, *A unified view of kernel k-means, spectral clustering and graph cuts*. Computer Science Department, University of Texas at Austin, 2004.
[39] R. Johansson, "System modeling and identification," 1993.
[40] J. W. Demmel, O. A. Marques, B. N. Parlett, and C. Vömel, "Performance and accuracy of lapack's symmetric tridiagonal eigensolvers," *SIAM Journal on Scientific Computing*, vol. 30, no. 3, pp. 1508–1526, 2008.
[41] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
[42] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.
[43] M. A. Porter, J.-P. Onnela, and P. J. Mucha, "Communities in networks," *Notices of the AMS*, vol. 56, no. 9, pp. 1082–1097, 2009.
[44] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 066111, 2004.
[45] M. Rosvall and C. T. Bergstrom, "Maps of information flow reveal community structure in complex networks," Citeseer, Tech. Rep., 2007.
[46] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.
[47] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *Science*, vol. 328, no. 5980, pp. 876–878, 2010.
[48] L. Massoulié, "Community detection thresholds and the weak ramanujan property," in *Proc. Ann. ACM Symp. Theory of Computing*, 2014, pp. 694–703.

[49] C.-S. Chang, W. Liao, Y.-S. Chen, and L.-H. Liou, "A mathematical theory for clustering in metric spaces," *IEEE Trans. Network Science and Engineering*, vol. 3, no. 1, pp. 2–16, 2016.
[50] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.
[51] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.
[52] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. Ann. ACM-SIAM symp. Discrete Algorithms*, 2007, pp. 1027–1035.
[53] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *Proceedings of the VLDB Endowment*, vol. 5, no. 7, pp. 622–633, 2012.
[54] R. Jaiswal and N. Garg, "Analysis of k-means++ for separable data," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2012, pp. 591–602.
[55] M. Agarwal, R. Jaiswal, and A. Pal, "k-means++ under approximation stability," in *Theory and Applications of Models of Computation*. Springer, 2013, pp. 84–95.
[56] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.
[57] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Comput. Commun.*, vol. 41, pp. 1–10, 2014.
[58] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.
[59] J. Kunegis and A. Lommatzsch, "Learning spectral graph transformations for link prediction," in *Proc. ACM Int. Conf. Machine Learning (ICML'09)*, June 2009, pp. 561–568.
[60] Y.-Y. Lo, W. Liao, and C.-S. Chang, "Temporal matrix factorization for tracking concept drift in individual user preferences," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 1, pp. 156–168, 2018.
[61] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu, "A new user similarity model to improve the accuracy of collaborative filtering," *Knowledge-Based Systems*, vol. 56, pp. 156 – 166, 2014.
[62] P. Erdös and A. Rényi, "On random graphs," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.
[63] A. Saade, F. Krzakala, and L. Zdeborová, "Spectral clustering of graphs with the bethe hessian," in *Proc. Ann. Conf. Advances in Neural Information Processing Systems*, 2014, pp. 406–414.
[64] L. Z. Aurelien Decelle, Florent Krzakala and P. Zhang. (2012) Mode-net: Modules detection in networks. [Online]. Available: $http://mode_net.krzakala.org/$
[65] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla, "Evaluating link prediction methods," *Knowledge and Information Systems*, vol. 45, no. 3, pp. 751–782, 2015.
[66] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on opportunistic forwarding algorithms," *IEEE Trans. on Mobile Computing*, vol. 6, no. 6, pp. 606–620, Jun. 2007.
[67] L. Isella, S. Juliette, A. Barrat, C. Cattuto, J. Pinton, and W. Van den Broeck, "What's in a crowd? analysis of face-to-face behavioral networks," *J. Theoretical Biology*, vol. 271, no. 1, pp. 166–180, 2011.
[68] N. Eagle and A. (Sandy) Pentland, "Reality mining: Sensing complex social systems," *Personal Ubiquitous Comput.*, vol. 10, no. 4, pp. 255–268, Mar. 2006.
[69] T. Wang, J. Qiu, S. Fu, and W. Ji, "Distributed fuzzy H_∞ filtering for nonlinear multirate networked double-layer industrial processes," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 5203–5211, 2017.
[70] T. Wang, J. Qiu, H. Gao, and C. Wang, "Network-based fuzzy control for nonlinear industrial processes with predictive compensation strategy," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

Tsunghan Wu received the BS degree in computer science and MS degree in electronics engineering from the National Taiwan University, Taipei, Taiwan, in 2006 and 2009, respectively. He is currently working toward the PhD degree in computer science at National Taiwan University. His research interests are in evolving networks, predictive models, and data analysis.

Cheng-Shang Chang (S'85-M'86-M'89-SM'93-F'04) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1983, and the M.S. and Ph.D. degrees from Columbia University, New York, NY, USA, in 1986 and 1989, respectively, all in electrical engineering.

From 1989 to 1993, he was employed as a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. Since 1993, he has been with the Department of Electrical Engineering, National Tsing Hua University, Taiwan, where he is a Tsing Hua Distinguished Chair Professor. He is the author of the book Performance Guarantees in Communication Networks (Springer, 2000) and the coauthor of the book Principles, Architectures and Mathematical Theory of High Performance Packet Switches (Ministry of Education, R.O.C., 2006). His current research interests are concerned with network science, big data analytics, mathematical modeling of the Internet, and high-speed switching.

Dr. Chang served as an Editor for Operations Research from 1992 to 1999, an Editor for the *IEEE/ACM TRANSACTIONS ON NETWORKING* from 2007 to 2009, and an Editor for the *IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING* from 2014 to 2017. He is currently serving as an Editor-at-Large for the *IEEE/ACM TRANSACTIONS ON NETWORKING*. He is a member of IFIP Working Group 7.3. He received an IBM Outstanding Innovation Award in 1992, an IBM Faculty Partnership Award in 2001, and Outstanding Research Awards from the National Science Council, Taiwan, in 1998, 2000, and 2002, respectively. He also received Outstanding Teaching Awards from both the College of EECS and the university itself in 2003. He was appointed as the first Y. Z. Hsu Scientific Chair Professor in 2002. He received the Merit NSC Research Fellow Award from the National Science Council, R.O.C. in 2011. He also received the Academic Award in 2011 and the National Chair Professorship in 2017 from the Ministry of Education, R.O.C. He is the recipient of the 2017 IEEE INFOCOM Achievement Award.

Wanjiun Liao (S'96-M'97-SM'06-F'10) received her Ph.D. degree in Electrical Engineering from the University of Southern California, USA, in 1997. She is a Distinguished Professor of Electrical Engineering Department, National Taiwan University (NTU), Taipei, Taiwan, where she was the Department Chair. She is the Director General of the Engineering and Technologies Department, Ministry of Science and Technology (MOST), Taiwan, and also an Adjunct Research Fellow of the Research Center for Information Technology Innovation, Academia Sinica, Taiwan. Her research is focused on the design and analysis of wireless networking, green communications, cloud networking and network virtualization.

Dr. Liao was an Associate Editor of IEEE Transactions on Wireless Communications and IEEE Transactions on Multimedia, and is on the Steering Committee of the IEEE Transactions on Mobile Computing. She was an IEEE Communications Society (ComSoc) Distinguished Lecturer, IEEE ComSoc Fellow Evaluation Committee, and IEEE Fellow Committee. She helped organize many IEEE conferences, including serving as the symposium Co-Chair of the IEEE GLOBECOM and the IEEE ICC, and the TPC CoChair of the IEEE VTC 2010 Spring and the IEEE PIMRC 2015. She received many awards and recognitions from government and different organizations. She is a Fellow of the IEEE.