Dynamic Social Influence Analysis through Time-dependent Factor Graphs

Chi Wang†, Jie Tang‡, Jimeng Sun§, Jiawei Han†

†University of Illinois at Urbana-Champaign, USA ‡Tsinghua University, China
§ IBM TJ Watson Research Center, USA

E-mail: {chiwang1,hanj}@illinois.edu, jietang@tsinghua.edu.cn, jimeng@us.ibm.com

Abstract

Social influence, the phenomenon that the actions of a user can induce her/his friends to behave in a similar way, plays a key role in many (online) social systems. For example, a company wants to market a new product through the effect of "word of mouth" in the social network. It wishes to find and convince a small number of influential users to adopt the product, and the goal is to trigger a large cascade of further adoptions. Fundamentally, we need to answer the following question: how to quantify the influence between two users in a large social network?

To address this question, we propose a pairwise factor graph (PFG) model to model the social influence in social networks. An efficient algorithm is designed to learn the model and make inference. We further propose a dynamic factor graph (DFG) model to incorporate the time information. Experimental results on three different genres of data sets show that the proposed approaches can efficiently infer the dynamic social influence. The results are applied to the influence maximization problem, which aims to find a small subset of nodes (users) in a social network that could maximize the spread of influence. Experiments show that the proposed approach can facilitate the application.

1 Introduction

With the success of many large-scale online social networks, such as Facebook and Twitter, the social networks are playing a very important role as a medium for the spread of information, ideas, and influences. In social networks, with the power of "word of mouth", a new idea or innovation can influence a large population in a very short period, but may also die out quickly. To understand the underlying dynamics of the social networks, it is very important to know how people influence with each other.

Social influence analysis has attracted a lot of interests from both the sociology and the data mining research communities. For example, Domingos and Richardson [7, 14] first propose the the influence maximization problem, in which the goal is to find a few "influential" members of the network. Kempe et al. [10] formalize the problem in discrete optimization and propose three cascade models for influence propagation. However, in these works, the influence between individual users is supposed to be available, which is not the case in most applications. Some other works study the relationships between social influences and correlation [1] or similarity [6]. However, they only present qualitative findings about social influences, but do not provide a quantitative measure of the influential strength. Tang *et al.* [17] presents a method to measure the influential strength. It formalizes the problem of social influence analysis as identifying which user has the highest probability to influence another user in the social network. Goyal *et al.* [9] learns the influence propagation probaility from action log data, which are often not publicly available. Also, similar behaviors do not always indicate friendship [13], although the network structure can be inferred with some assumptions of the information diffusion model [15]. When action data are unavailable or insufficient, it remains a question how to infer arbitrary pairwise influence varying with time, given the dynamic topological network structures.

Motivating Application

To clearly motivate this work, we demonstrate with an example of coauthor network. In Figure 1, the left figure shows an example of coauthor network at different time windows t, $t+1$, and $t+2$, in which the networks (structures and contents) change along with the time. The dynamic social influence analysis takes the social networks at different time windows as input and outputs the dynamic social influential strengths between individual users (as shown in the middle figure, where the social network is the accumulation of the networks at different time windows). For each social relationship, e.g., the relationship between Frank and Carol, the analysis generates two directed edges, each associated with a vector of influence scores $\boldsymbol{\mu} = \{\mu^t, \mu^{t+1}, \mu^{t+2}\}$. With the generated dynamic social influence scores, we can consider many important applications such as influence maximization, expert finding, and social recommendation. The right figure gives an example of application to influence maximization, where the goal is to identify the

978-0-7695-4375-8/11 $26.00 © 2011 IEEE
DOI 10.1109/ASONAM.2011.116

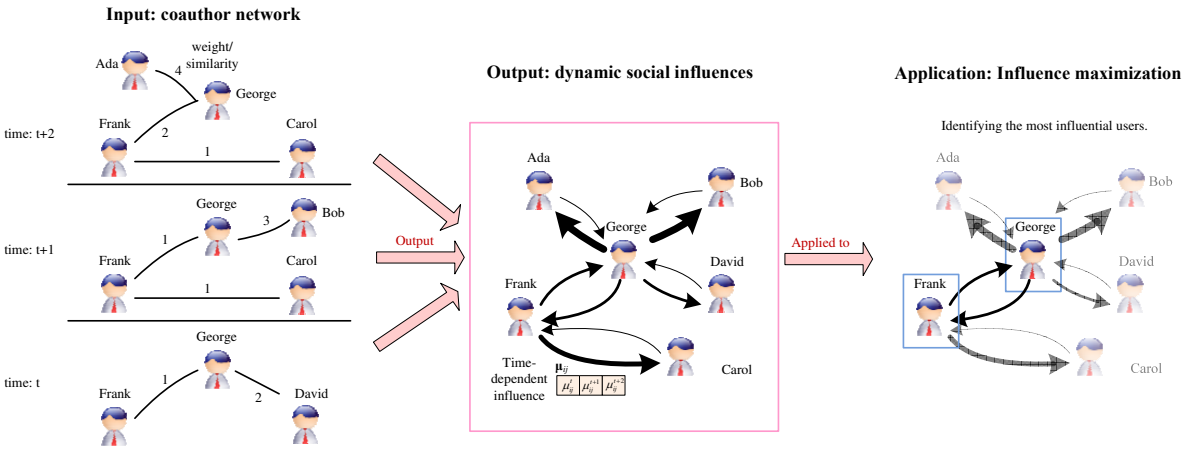


Figure 1: Example of dynamic social influence analysis on the co-author network.

most influential users. Based on the computed social influences, one can use algorithms developped by previous researchers([10, 4]) to find the subset of users (e.g., George and Frank) who could maximize the spread of influence. The problem to study in this paper is how to efficiently and effectively model the dynamic social influence for real large networks.

Contributions

The problem of dynamic social influence analysis is quite different from existing works on social network analysis. First, we mainly use the topological structures to model the pairwise influence, so as to analyze the influence when the action log data are unavailable or uneffective. Second, we notice that social influences are highly time-dependent. The influence of a user on another depends on their interactions in previous time windows.

In this paper, we formulate and tackle the problem of dynamic social influence analysis, and present a pairwise factor graph (PFG) model to model the pairwise influence. Specifically, the influence between two users is modeled as a marginal probability of two hidden variables in the factor graph model. An efficient learning algorithm is proposed. Next, we propose a time-dependent factor graph (DFG) model to further incorporate the time information, which is described as a factor function across time windows. Thus influence is propagated across social networks of different time windows.

2 Problem Formulation

In this section, we present the problem formulation and define notations used throughout this paper.

In general, the input of dynamic social influence analysis is a serial of time-dependent social networks $\{G^t\} = \{(V^t, E^t)\}$, where V^t is a set of nodes (users, entities) appearing within the time window t and E^t is the set of directed/undirected edges. Each edge $e_{ij}^t \in E^t$ is associated with a weight/similarity w_{ij}^t, which can be defined in different ways, depending on the specific application. In the following of this section, we first introduce some terminology, and then define the dynamic social influence analysis problem.

Pairwise social influence: Pairwise social influence from user v_i to v_j denoted as μ_{ij} is a numerical weight associated with the edge e_{ij}. In most cases, the social influence score is asymmetric, i.e., $\mu_{ij} \neq \mu_{ji}$.

Dynamic social influence: Dynamic social influences from user v_i to v_j is defined as a vector of numerical weights $\boldsymbol{\mu}_{ij}$ associated with the edge e_{ij}. The t-th component $\mu_{ij}^t \in \boldsymbol{\mu}_{ij}$ is the social influence of user v_i on v_j at time t. The dynamic social influence is also asymmetric.

Based on the above concepts, we can define the tasks of dynamic social influence analysis. Given a serial of time-dependent social networks $\{G^t\} = \{(V^t, E^t)\}$, the expected output of dynamic social influence is the dynamic social influence scores for each social tie.

Problem: Dynamic social influence analysis. Given T time-dependent social networks $\{G^t\} = \{(V^t, E^t)\}_{t=1}^T$, where V^t is the set of users appearing at the time-window t and E^t is the set of directed/undirected edges between these users, and each edge $e_{ij}^t \in E^t$ is associated with a weight/similarity w_{ij}^t, how to find the dynamic social influence network $G' = (V, E, \Omega)$? Here the nodes and edges of G' are the accumulation of the time-dependent networks, i.e., $V = V^1 \cup V^2 \cdots \cup V^T$ and $E = E^1 \cup E^2 \cdots E^T$, Ω is the set of dynamic social influences, i.e., $\{\boldsymbol{\mu}_{ij}\}$.

The obtained dynamic social influence scores can be applied to many applications. As a case study, we apply it to a state-of-the-art problem in social network analysis: influence maximization [7, 14, 10].

Application: Influence maximization. Given a social network with the social influence on each social relationship, i.e. $G' = (V, E, \Omega)$, how to find a small subset of nodes (seed nodes) from the network that could maximize the spread of influence.

The problem of influence maximization has been proven to be a NP-hard problem [10]. A greedy approximation algorithm can guarantee that the influence spread is no worse than $(1 - 1/e)$ of the optimal influence spread. One major problem of the greedy algorithm is its low efficiency. Chen *et al.* have developped new heuristics [5, 4] to accelerate the greedy algorithm. The dynamic social influences learnt by our method can provide the input of these algorithms.

3 Our Approach

Based on the input weighted network, we formalize the social influence problem in a factor graph model. Our main idea is to use a marginal probability to define the pairwise influence between users. Other social information, i.e., users' attribute information, social similarities/weights, and network structures are captured via different types of factor functions, which form the basic components of the factor graph model. By learning and inferring with the factor graph model, we can obtain all the marginal probabilities. For ease of explanation, we first describe the pairwise factor graph (PFG) model without considering the time information, which will be further discussed in Section 3.3.

3.1 Pairwise Factor Graph (PFG) Model

Now we formally define the proposed PFG model.

Variables The PFG model has the following components: a set of observed variables $V = \{v_i\}_{i=1}^N$, which corresponds to the N users in the input network, and a set of hidden variables $Y = \{y_i\}_{i=1}^N$, representing which node the user v_i randomly chooses to follow when selecting an activity. v_i can follow the activity of his neighbors when $y_i \in NB(i)$, or create an activity himself when $y_i = i$. So $y_i \in SC(i) = NB(i) \cup \{i\}$, where $NB(i)$ is the indices set of neighboring nodes of v_i. v_i will follow an action from v_{y_i} with the probability $P(y_i|V)$. So we use $P(y_i|V)$ to reflect the influence of v_{y_i} on v_i.

This activity selection scheme is similar with that in [6]. But we differentiate neighbors' influence explicitly via the activity selection probability rather than presume $P(y_i|V)$ a uniform distribution. We assume the influence does not vary with time in PFG model. And we will address the dynamic change of influence in next section.

Figure 2 shows a simple example of an PFG. The observed data consist of four users $\{v_1, \ldots, v_4\}$, which have corresponding hidden variables $Y = \{y_1, \ldots, y_4\}$. The edges between the hidden nodes indicate the four social relationships in the original network (a.k.a. the edges of the input network).

Factor functions We define two types of factor functions to capture the social information.

- **Node factor function** $g(y_i|V)$ is a factor function defined on node (user) v_i. It is used to describe the local (user-specific) information.
- **Edge factor function** $f(y_i, y_j|V)$ is a factor function defined on the edge of the input network. It can be used to describe the dependencies between users.

Generally, the two factor functions can be defined in various ways. In this work, we define the node factor function g as:

$$g(y_i|V) = \begin{cases} \frac{w_{iy_i}}{\sum_{j \in NB(i)}(w_{ij}+w_{ji})} & y_i \neq i \\ \frac{\sum_{j \in NB(i)} w_{ji}}{\sum_{j \in NB(i)}(w_{ij}+w_{ji})} & y_i = i \end{cases} \quad (1)$$

The node factor function g reflects the likelihood that one user is directly influenced by another solely regarding the weight of the edge connecting both. The edge weight can be defined based on similarity or interaction, determined by the application. It reflects the intuition that if v_i has a high similarity or interaction with v_{y_i}, then v_{y_i} may have a high influence on node v_i. One can also extend the function g by defining the weight according to other node-specific information such as social roles, personal attributes, and ascribed relationships.

The edge factor function f is used to describe the dependencies between users, for example, how likely two friends are influenced by the same user. Formally, the edge factor function is defined as follows:

$$f(y_i, y_j|V) = \begin{cases} \frac{u}{|SC(i) \cap SC(j)|} & y_i = y_j \\ \frac{1-u}{|SC(i)||SC(j)| - |SC(i) \cap SC(j)|} & y_i \neq y_j \end{cases} \quad (2)$$

where u is a propagation weight, ranging in $[0, 1]$, indicating the probability of two connected users influenced by the same user. Intuitively, u characterizes the preference of the model for social propagation (indirect influence). If set $u = 0$, then there is strong resistence towards social propagation. Setting $u = 1$ results in a model of thorough propagation.

For the definitions above we notice that the node factor function relies on the neighboring information of one node, while the edge factor function relies on that of two linked nodes. This is formally indicated by $g(y_i|V) = g(y_i|\{v_k\}_{k \in SC(i)})$ and $f(y_i, y_j|V) = f(y_i, y_j|\{v_k\}_{k \in NB(i) \cup NB(j)})$.

Theoretically, one can incorporate any types of features into the PFG model. For example, one can even define some useful constraints like: the old men will hardly influence the young generation (users with age< 18) on the actions of buying fashion clothes. One can also define more complex features over triangle (three user) or quadrangle (four user).

Probabilistic influence A factor graph model is constructed based on the above defined components. Then for a social relationship e_{ij}, the influence of v_j on v_i can be defined as a marginal probability:

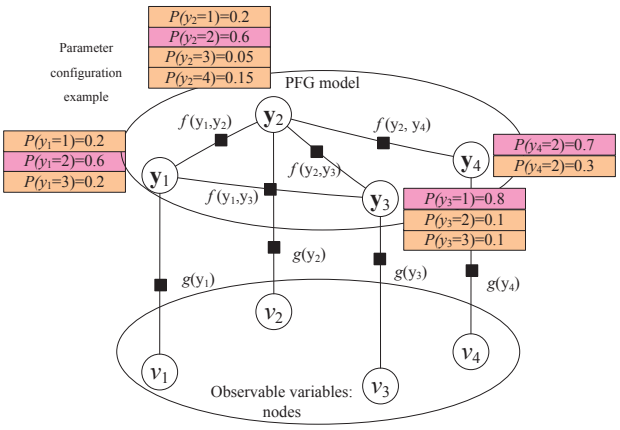


Figure 2: Graphical representation of the pairwise factor graph. $\{v_1, \ldots, v_4\}$ are observable nodes in the social network; $\{y_1, \ldots, y_4\}$ are hidden variables defined on all nodes; $g(.)$ represents a feature function defined on a node and $f(.)$ represents a feature function defined on a social relationship.

$$P(y_i = j|V) = \sum_{\sim\{y_i\}} P(y_1, \ldots, y_i = j, \ldots, y_N|v_1, \ldots, v_N) \tag{3}$$

with

$$P(y_1, \ldots, y_N|v_1, \ldots, v_N) = \frac{1}{Z} \prod_{v_i \in V} g(y_i|V) \prod_{e_{ij} \in E} f(y_i, y_j|V) \tag{4}$$

where $P(y_1, \ldots, y_N|v_1, \ldots, v_N)$ is the joint probability of all hidden variables in the PFG model , and Z is a normalizing factor.

In this way, the problem of influence analysis is formalized as calculating the marginal probabilities conditioned on the observed graph $\{P(y_i = j|V)\}_{j \in SC(i)}$ for each node v_i. Since V are observable nodes, $P(y_i)$ is proportional to $P(y_i|V)$. So from now on we omit the observed variables V for notation convenience. The marginal probability $P(y_i = j)$ characterizes the influence score of v_j on v_i. One parameter configuration is shown in Figure 2. For example, the probabilities $\{P(y_1 = j)\}_{j=1,2,3}$ suggests that node v_2 has the largest influence on v_1. Also we can see v_2 has a high self-influence.

3.2 Model Learning and Inference

To infer the social influences, we need to approximate all marginal probabilities $\{P(y_i = j)\}$.

The joint probability function can be represented as a factor graph model because it is a product of factor functions g and f. Factor graph is a bipartite graph comprised of two kinds of nodes: variables and functions. To differentiate the function nodes on the graph, we denote g_i for the g function defined on y_i and f_{ij} for f defined on y_i, y_j. When a variable appears in the definition of a function, there is one edge between the variable node, in our case y_i, and the function node, in our case f_{i*}, f_{*i} or g_i.

In sum-product algorithm [12], messages are passed between neighboring variable node and function node. Message passing is initiated at the leaves. Each node y_i remains idle until messages have arrived on all but one of the edges incident on the node y_i. Once these messages have arrived, node y_i is able to compute a message to be sent onto the one remaining edge to its neighbor. After sending out a message, node y_i returns to the idle state, waiting for a "return message" to arrive from the edge. Once this message has arrived, the node is able to compute and send messages to each of neighborhood nodes. The calculation terminates when two messages have passed on every edge. At each variable node, the product of all incoming messages is its marginal probability.

However, traditional sum-product algorithm cannot be directly applied to PFG because it may contain cycles. Some nodes will always stay idle if we use the same message passing scheme. One solution is a procedure known as *junction tree algorithm* [2] for exact inference. The junction tree is a tree-structured undirected graph generated from arbitrary triangulated dependency graph, and can be solved by sum-product. Nevertheless, the computational cost of the algorithm is determined by the number of variables in the largest clique and will grow exponentially with this number in the case of discrete variables. To reduce the computational cost, we can do approximate inference instead of exact inference. A widely used method *loopy belief propagation* (LBP) [8] simply applies the sum-product algorithm in a cycle-containing graph. It passes message iteratively with flooding schedule.

New SIP Inference and Learning Algorithm The message passing method above has two disadvantages: 1) the sum-product algorithm requires that each node need wait for all(-but-one) message to arrive; 2) the messages sent between nodes does not directly reflect the influence between users, hence the internal messages are not necessary for influence analysis after the iteration.

To deal with these problems, we propose a social influence propagation (SIP) algorithm, in which we utilize the particular structure of our graphical model to simplify the message passing process, and introduce a three-dimensional influence message Λ to replace the original message m, and thus convert the message passing update rules into a new update rule of Λ. We let $\Lambda_{ijk} = m_{f_{ij} \to y_j}(k)$. Thus the messages passed from y_i to y_j through f_{ij} are now recorded in $\{\Lambda_{ijk}, k \in SC(j)\}$. The introduced variables Λ have the following explanation. Message Λ_{ijk} reflects, from the perspective of node v_j, considering his/her relationship with v_i, how likely node v_j think he/she is influenced by node v_k. Λ_{ijk} only exists when v_i is a neighbor of v_j and v_k is either v_j's neighbor or v_j itself. In each iteration, every element Λ_{ijk} is updated (in arbitrary order) with the following update rule.

Input: $G = (V, E)$
Output: $G' = (V, E, \Omega)$

Calculate the node feature function $g(v_i, y_i)$;
Calculate b_{ij} according to Eq. 8;
Initialize all $\mathbf{\Lambda}_{ijk} \leftarrow 0$;
repeat
  foreach *edge* (i, j) **do**
    foreach *neighboring node* $k \in SC(j)$ **do**
      Update Λ_{ijk} according to Eq. 5;
    end
  end
until *convergence*;
foreach *node* v_j **do**
  foreach $k \in SC(j)$ **do**
    Compute μ_{kj} according to Eq. 7
  end
end

Generate $G' = (V, E, R)$ according to $\{\mu_{ij}\}$

Algorithm 1: The new SIP inference algorithm.

$$\Lambda_{ijk} = \frac{1}{Z_i} \sum_{l \in SC(i)} f_{ij}(l,k) b_{il} \prod_{s \in NB(i) \setminus \{j\}} \Lambda_{sil} = \sum_{l \in SC(i)} f_{ij}(l,k) \frac{\mu_{li}}{\Lambda_{jil}} \quad (5)$$

where $\mu_{ji} = P(y_i = j)$ is the probabilistic influence of node v_j on node v_i, and can be decomposed as the product of messages towards v_i about $y_i = j$,

$$\mu_{ji} = \frac{1}{Z_i} b_{ij} \prod_{s \in NB(i)} \Lambda_{sij} \quad (6)$$

$$= \frac{b_{ij} \prod_{s \in NB(i)} \Lambda_{sij}}{\sum_{k \in SC(i)} b_{ik} \prod_{s \in NB(i)} \Lambda_{sik}} \quad (7)$$

Here Z_i is a normalization factor; b_{ij} is a normalized node feature function defined as follows.

$$b_{ij} = \frac{g_i(y_i = j)}{\sum_{k \in SC(i)} g_i(y_i = k)} \quad (8)$$

The normalization is needed in generic case, although our definition of g_i already ensures $\sum_{k \in SC(i)} g_i(y_i = k) = 1$.

Now we do not need to calculate the actual messages sent between variables and functions on the factor graph. Instead, we only need to update Λ according to Eq. 5. The order-independent property of the new update rule offers the basis of parallelization. The algorithm is summarized in Algorithm 1.

The weight u of social propagation can be assigned according to the network property. In this work we use an alternate optimization algorithm (a.k.a. EM, as summarized in Algorithm 2) to maximize the joint probability (4). This algorithm can learn the the parameter u, as well as the "representative" v_{y_i} for every node v_i.

3.3 Dynamic Factor Graph (DFG) Model

Now we describe how to incorporate the time information into the PFG model, which results in the DFG model. Given T time-specific social networks $\{G^t\} = \{(V^t, E^t)\}_{t=1}^T$, DFG models the networks as a sequence of time-dependent factor graphs. At each time window, the factor graph has a similar structure with the PFG model. In addition, each factor graph also depends on the factor graph of the previous time window. Thus the sequence of time-dependent factor graphs forms a Markov chain. Factor functions are defined between the variables of two consecutive time-dependent factor graphs. A forward-backward message passing process is then designed to capture the dependencies between networks of two time windows. Figure 3 shows the graphical representation of the DFG model.

Input: $G = (V, E, W)$
Output: $u, R = \{y_i\}$

Initialize $u = 0.5$;
repeat
  E-step: **begin**
    Execute SIP on G, but replacing the sum with max in the update rule;
    foreach *node* v_j **do**
      Compute $y_j \leftarrow arg\max_{i \in SC(j)} \mu_{ij}$
    end
    $R \leftarrow \{y_1, \ldots, y_N\}$;
  end
  M-step: optimize the log likelihood of Eq. 4, with the following gradient: $\frac{\partial L}{\partial u} = F(Y = R) - E_{p(Y|u)} F(Y)$, where $F(Y) = \sum_{v_i \in V} \log g(y_i) + \sum_{e_{ij} \in E} \log f(y_i, y_j)$.
until *convergence*;

Algorithm 2: The EM algorithm to learn u and representative indices $\{y_i\}$.

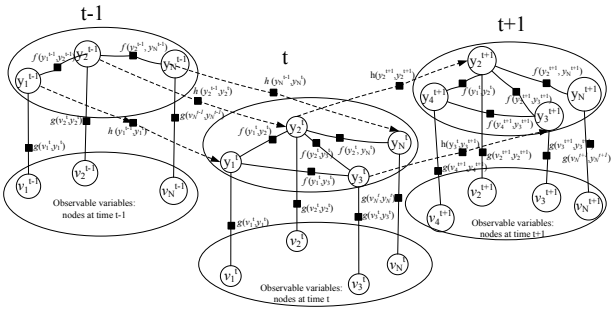


Figure 3: Graphical representation of the dynamic factor graph.

To formally define the DFG model, we add a superscript t to the variables (v_i, y_i, Λ_{ijk}, and μ_{ij}) in PFG. Within the factor graph of each time window, the factor functions (node factor function and edge factor function) are defined similarly to the PFG model. Between two consecutive time-dependent factor graphs, we define a bridge factor function between y_i^t and y_i^{t+1}. In this way, the influence μ_{ji}^t (or $P(y_i^t = j)$) of user v_j on v_i not only is determined by their local and network structure information at time t, but also depends on their historic influence. Specifically, the bridge factor function as:

$$h(y_j^t, y_j^{t+1}) = \begin{cases} \frac{q}{|SC(j)|} & y_j^t = y_j^{t+1} \\ \frac{1-q}{|SC(j)|(|SC(j)|-1)} & y_j^t \neq y_j^{t+1} \end{cases} \quad (9)$$

where $q \in [0, 1]$ is a weight, indicating the probability of one user's influence on another preserves with time chang-

ing. The weight q captures the time-dependencies for the dynamic social influence analysis. It can be learned in a similar way as Algorithm 2.

Now the time-dependent probabilistic influence $P(y_i^t = j, v_i^t)$ is similar to Eq. 3, except that the joint probability is replaced with:

$$P(y_1^t, \ldots, y_N^t, v_1, \ldots, v_N) = \prod_{v_i^t \in V^t} g(y_i^t, v_i^t) \prod_{e_{ij} \in E^t} f(y_i^t, y_j^t) \prod_{v_i^t \in V} h(y_i^{t-1}, y_i^t) \quad (10)$$

We generalize SIP learning algorithm to solve the learning problem for the DFG model. By introducing two new variables α and β, respectively representing the forward and backward messages passed between two time-windows, we can obtain the following update rules:

$$\Lambda_{ijk}^t = \frac{1}{Z_i^t} \sum_{l \in SC(i)} f_{ij}(l,k) b_{il}^t \prod_{s \in NB(i) \setminus \{j\}} \Lambda_{sil}^t \sum_{k' \in SC(i)} h(k',k) \alpha_{k'i}^{t-1} \sum_{k'' \in SC(i)} h(k'',k) \beta_{k''i}^{t+1} \quad (11)$$

$$\alpha_{ki}^t = \frac{1}{Z_i^t} \sum_{l \in SC(i)} h(l,k) \alpha_{li}^{t-1} b_{il}^t \prod_{s \in NB(i)} \Lambda_{sil}^t \quad (12)$$

$$\beta_{ki}^t = \frac{1}{Z_i^t} \sum_{l \in SC(i)} h(l,k) \beta_{li}^{t+1} b_{il}^t \prod_{s \in NB(i)} \Lambda_{sil}^t \quad (13)$$

$$Z_j^t = \sum_{l \in SC(j)} \alpha_{lj}^{t-1} \beta_{lj}^{t+1} b_{jl} \prod_{i \in NB(j)} \Lambda_{ijl}^t \quad (14)$$

Thus the probabilistic influence can be calculated by

$$\mu_{kj}^t = \frac{1}{Z_j^t} \alpha_{kj}^{t-1} \beta_{kj}^{t+1} b_{jk} \prod_{i \in NB(j)} \Lambda_{ijk}^t \quad (15)$$

where α_{kj}^{t-1} is the message from the previous time window and β_{kj}^{t+1} is the message from the next time window. Above is the solution when influence scores in all time windows are calculated together. Given that the influence is evolving forward with time, which means $\{\mu_{ij}^t\}$ should be generated without information about $\{\mu_{ij}^{t+1}\}$, we can constraint the message passing process by only allowing forward message α and discarding β. Then it turns out we can compute the influence time by time, with a forward version of SIP.

Both the forward-backward and forward version of the SIP algorithm inherit the nice property of "local" update, which makes the algorithm easy to be parallelized.

4 Experimental Results

In this section, we present various experiments to evaluate the efficiency and effectiveness of the proposed approach.

4.1 Experiment Setup

We perform experiments on three real-world data sets: two of them are coauthor network (Coauthor) and citation network (Citation) extracted from an online academic search system Arnetminer[1]; the third (called Film) is crawled from Wikipedia under the category of "English-language films"[2], , comprised of films, directors, actors, and writers. We use them to evaluate the efficiency of our approach and how the learned dynamic social influences can help other social networking application.

Table 1: Scalability performance on real data sets.

Data Set	#Node	#Edge	Density	SIP	JuncT
Citation	127K	374K	10^{-5}	20m	N/A
Coauthor	61K	152K	10^{-3}	336s	2.06h
Film	34K	142K	10^{-2}	208s	52.9m

We makes use of the topic information on each node to define the weight/similarity for every edge. Specifically, each node is associated with a vector $\theta_v \in \mathbf{R}^T$ of T-dimensional topic distribution $\sum_z \theta_{vz} = 1$. Each element θ_{vz} is the probability (importance) of the node on topic z. Thus the weight between users is defined as $w_{ij} = \alpha_{ij} \theta_j^z$, where m_{ij} is the number of interaction times (e.g., coauthored papers) by v_i and v_j. The topic information is extracted using a statistical topic model [3].

We set the maximum number of iterations as 500 and the convergence threshold for the variation of μ to $1e-3$. The algorithm can quickly converge after 60-100 iterations in most of the times.

Table 1 lists the CPU time required for estimating the social influence on the three data sets. The new algorithm (SIP) is much faster (> 10 times faster) than the sum-product + junction tree algorithm. When the social network is large, e.g., the citation network, the junction tree does not work due to memory limitation.

4.2 Case Study

We present several case studies from the Coauthor data set to demonstrate the effectiveness of the proposed approaches.

Pairwise influence Table 2 shows an example of pairwise influences generated by our approach. The influence results show some interesting patterns. For example, some junior researcher (e.g., Chao Liu) may be mainly influenced by his advisor; while the established researcher (e.g., Jiawei Han) may be mainly influenced by his own opinion, but will also be influenced by some rising star (e.g., Xifeng Yan). This kind of result can help identifying relationship or differentiating social roles.

Most influential users Table 3 lists the most influential researchers in the data mining field in 2009. Each researcher is scored by accumulating their influences on the other researchers. We see that the most influential researchers are consistent with their expertise (not all researchers are included in our data set).

[1] http://arnetminer.org

[2] http://en.wikipedia.org/wiki/Category:English-language_films

Table 2: Influence between Jiawei Han and his coauthors. The number, e.g., from one coauthor to Jiawei Han, indicates the percentage of the influence in the total influences of all coauthors on Jiawei Han.

Coauthor	Jiawei Han on Coauthor	Coauthor on Jiawei Han
David Clutter	99.85%	0.07%
Hwanjo Yu	91.66%	0.87%
Chao Liu	86.67%	0.63%
Xifeng Yan	79.00%	3.28%
Micheline Kamber	78.70%	0.45%
Krzysztof Koperski	76.71%	1.30%
Yongjian Fu	72.81%	1.49%
Bin He	70.79%	0.15%

Table 3: Example of the most influential users on the "data mining" topic discovered from the Coauthor data set.

Jiawei Han, Heikki Mannila, Christos Faloutsos, Philip S. Yu, Hans-Peter Kriegel, Vipin Kumar, Dimitrios Gunopulos, C. Lee Giles, Ming-Syan Chen, Shusaku Tsumoto, Michael R. Berthold, Chengqi Zhang, Wei Wang, Ronen Feldman

Dynamic influences We further conduct a dynamic influence analysis. We use Dr. Jian Pei as the example to analyze how the influences of Dr. Pei on or by his coauthors change during 2000 and 2009. Table 4 shows the dynamic analysis result. We see that the influence evolution uncovers the growing up of Dr. Pei. For example in 2000, Dr. Pei is mainly influenced by Prof. Han, while he only has limited influence on Prof. Han. After 2004, Dr. Pei starts influencing some other researchers (e.g., Chun Tang and Shiwei Tang). While in 2008, Dr. Pei already becomes a mature researcher and has many strong influences on other researchers.

4.3 Applications

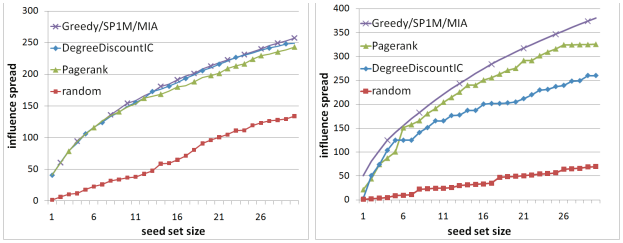
The dynamic social influence analysis can benefit many applications. We use the influence maximization problem as an example to demonstrate.

The influence maximization problem is to find a small subset of nodes (seed nodes) in a social network that could maximize the spread of influence [7, 14, 10]. In most previous work, different algorithms are evaluated under simple assumptions about pairwise influence. Now the output of our social influence analysis can be used as the input of the influence maximization problem, and we can test whether existing optimization algorithms perform as well as they do under the naive assumptions.

Figure 4a shows the solution found by several state-of-the-art algorithms when we define the spread probability from v_i to v_j simply as $\frac{1}{d_j}$ (referred as WC model), where d_j is the in-degree of v_j. Beyond Greedy algorithm, we also test SP1M [11], using a simplified ICM model and MIA [4], a heuristic algorithm for general ICM. Baseline algorithms include: 1) Random, randomly picking seeds, 2)PageRank, selecting nodes with top PageRank score and 3)DegreeDiscountIC, a heuristic algorithm with good performance in UICM[5]. Greedy algorithm provides best results as known. In WC model, SP1M and MIA perfectly match Greedy. DegreeDiscountIC has nearly matching performance (96.1%). They all beat the other 2 baselines. WC model presumes equal influence from all neighbors, while PFG does not. When we replace the weights of WC with PFG influence score, as shown in Figure 4b, SP1M and MIA still match Greedy optimum, while DegreeDiscountIC degrades its performance to 80% of PageRank performance, which is already worse than Greedy by 14.4%. Therefore, by applying our influence results, we find that SP1M and MIA are much better approximation than DegreeDiscountIC if the assumption of identical influence does not hold. PageRank is no better than DegreeDiscountIC in WC model but it is relatively stable in different models. These results imply that we can better distinguish the power of different algorithms with the influence mined from real data than with the simplified hypothetical influence.

Table 4: Dynamic influence analysis for Dr. Jian Pei during 2000-2009. Due to space limitation, we only list coauthors who most influence on/by Dr. Pei in each time window.

Year	Pairwise	Influence
2000 - 2001	Influence on Dr. Pei	Jiawei Han (0.4961)
	Influenced by Dr. Pei	Jiawei Han (0.0082)
2002 - 2003	Influence on Dr. Pei	Jiawei Han (0.4045), Ke Wang (0.0418), Jianyong Wang (0.019), Xifeng Yan (0.007), Shiwei Tang (0.0052)
	Influenced by Dr. Pei	Shiwei Tang (0.436), Hasan M.Jamil (0.4289), Xifeng Yan (0.2192), Jianyong Wang (0.1667), Ke Wang (0.0687)
2004 - 2005	Influence on Dr. Pei	Jiawei Han (0.2364), Ke Wang (0.0328), Wei Wang (0.0294), Jianyong Wang (0.0248), Philip S. Yu (0.0156)
	Influenced by Dr. Pei	Chun Tang (0.5929), Shiwei Tang (0.5426), Hasan M.Jamil (0.3318), Jianyong Wang (0.1609), Xifeng Yan (0.1458), Yan Huang (0.1054)
2006 - 2007	Influence on Dr. Pei	Jiawei Han (0.1201), Ke Wang (0.0351), Wei Wang (0.0226), Jianyong Wang (0.018), Ada Wai-Chee Fu (0.0125)
	Influenced by Jian Pei	Chun Tang (0.6095), Shiwei Tang (0.6067), Byung-Won On (0.4599), Hasan M.Jamil (0.3433), Jaewoo Kang (0.3386)
2008 - 2009	Influence on Dr. Pei	Jiawei Han (0.2202), Ke Wang (0.0234), Ada Wai-Chee Fu (0.0208), Wei Wang (0.011), Jianyong Wang (0.0095)
	Influenced by Dr. Pei	ZhaoHui Tang (0.654), Chun Tang (0.6494), Shiwei Tang (0.5923), Zhengzheng Xing (0.5549), Hasan M.Jamil (0.3333), Jaewoo Kang (0.3057)



(a) weight ← WC (b) weight ← PFG

Figure 4: Influence spread by different algorithms.

Table 5 presents the discovered seed nodes by three different schemes to set the cascade influence scores. For each set of seed nodes, we calculate the *density* measure in network science, dividing the sum of coauthor papers by the number of different pairs between seeds, i.e. $\frac{10\times9}{2} = 45$ in our case. The larger is the density, the more redundancy the seed nodes have in the network. To maximize the influence

Table 5: Discovered seed nodes in influence maximization by the greedy algorithm with different influence setting. *Unique*: influence=unique probability (0.01); *WC*: influence=inverse of in-degree; *PFG*: influence = result of PFG.

No.	Unique	WC	PFG
1	Philip S. Yu	Philip S. Yu	Jiawei Han
2	Jiawei Han	Jiawei Han	Qiang Yang
3	Christos Faloutsos	Wei Wang	Christos Faloutsos
4	Qiang Yang	Christos Faloutsos	Heikki Mannila
5	Heikki Mannila	Heikki Mannila	Vipin Kumar
6	Wei Wang	C. Lee Giles	C. Lee Giles
7	Jian Pei	Shusaku Tsumoto	Saso Dzeroski
8	Vipin Kumar	Jian Pei	Graham J. Williams
9	Bing Liu	Bing Liu	Myra Spiliopoulou
10	C. Lee Giles	Joost N. Kok	Eamonn J. Keogh
Density	0.4222	0.2444	**0.1778**

spread, it is desirable to minimize the density. We see that our approach clearly outperforms the other methods. It can be observed that Philip S. Yu and Jian Pei are not selected as a top-10 seed in PFG model due to the large overlap of the nodes influenced by them and by other seeds. In UICM and WC model, influence from neighbors to the node are independent, while in PFG, the correlation of influence between neighbors is captured.

5 Related Work

Much effort has been made for social network analysis (*e.g.*, [18]). As for social influence analysis, methods to qualitatively measure the existence of influence were proposed in [1, 16]. Crandall *et al.* [6] studied the correlation between social similarity and influence. Tang *et al.* [17] presented a method for measuring the influential strength. Action logs were used to learn the influence propagation probability [9] and the network structure [15]. To the best of our knowledge, no previous work has been conducted for quantitatively measuring the dynamic social influence without action log data.

Another line of related work is influence maximization. Domingos and Richardson [7, 14] first proposed the the influence maximization problem, in which the goal is to find a few "influential" members of the network. Kempe *et al.* [10] formalized the problem in discrete optimization and propose three cascade models for influence propagation. Chen *et al.* [5, 4] further proposed a heuristics-based method to improve the efficiency of influence maximization.

6 Conclusion

In this paper, we study the problem of dynamic social influence analysis. We propose a Pairwise Factor Graph (PFG) model to formalize the problem in probabilistic model, and we extend it by incorporating the time information, which results in the Dynamic Factor Graph (DFG) model. Experimental results on three different types of data sets demonstrate that the proposed approach can effectively discover the dynamic social influences. We apply the inferred social influence to help influence maximization. Parallelization of our algorithm can be done in future work to scalue it up further.

7 Acknowledgements

Research was sponsored in part by the U.S. National Science Foundation under grant IIS-09-05215, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 (NS-CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. Jie is supported by the NSFC (No. 61073073, No. 60703059), NSFC Key Fund (No. 60933013), National High-tech R&D Program (No. 2009AA01Z138).

References

[1] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *KDD'08*, pages 7–15, 2008.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD '10*, Washington D.C., July 2010.

[5] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD '09*, 2009.

[6] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *KDD'08*, pages 160–168, 2008.

[7] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD '01*, pages 57–66, 2001.

[8] B. J. Frey. *Graphical models for machine learning and digital communication*. MIT Press, Cambridge, MA, USA, 1998.

[9] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *WSDM '10*, 2010.

[10] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD '03*, pages 137–146, 2003.

[11] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In *PKDD*, pages 259–271, 2006.

[12] F. R. Kschischang, S. Member, B. J. Frey, and H. andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 2001.

[13] T. La Fond and J. Neville. Randomization tests for distinguishing social influence and homophily effects. In *WWW '10*, 2010.

[14] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD '02*, pages 61–70, 2002.

[15] M. G. Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *KDD '10*, 2010.

[16] P. Singla and M. Richardson. Yes, there is a correlation: - from social networks to personal behavior on the web. In *WWW'08*, pages 655–664, 2008.

[17] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *KDD '09*, 2009.

[18] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge: Cambridge University Press, 1994.