

# Advisor-advisee Relation Mining from Dynamic Collaboration Network

Chi Wang<sup>1</sup>  
chiwang1@illinois.edu

Jiawei Han<sup>1</sup>  
hanj@illinois.edu

Yuntao Jia<sup>1</sup>, Duo Zhang<sup>1</sup>  
{yjia3, yintao}@uiuc.edu

Yintao Yu<sup>1</sup>  
dzhang22@illinois.edu

Jie Tang<sup>2</sup>  
jietang@tsinghua.edu.cn

Jingyi Guo<sup>2</sup>  
guojy07@mails.tsinghua.edu.cn

<sup>1</sup> University of Illinois at Urbana-Champaign <sup>2</sup> Tsinghua University, Beijing China

## ABSTRACT

Information network contains abundant knowledge of the relations between people or entities. Such kind of knowledge, however, is often hidden in a network in which nodes are not explicitly distinguishable. For example, given a collaboration network of researchers, the advisor-advisee relationship is hidden in the collaboration records and characterized by the development of each researcher and the change of their social role. To discover this potential information from the seemingly homogeneous network, this paper takes academic network as a case study to distinguish the role of the nodes and analyze advisor-advisee relations between them. A time-constrained probabilistic factor graph model (TPFG) is proposed. It takes the DBLP collaboration network as input and models jointly the likelihood of one researcher advising another. An efficient algorithm is designed to estimate the optimal joint probability by message propagation on the whole graph. Based on the estimation we can rank the most probable advisors for every author. Experimental results show that the proposed approach can efficiently infer the advisor-advisee relationship and achieve a high accuracy over 80%. The methodology can be generalized.

## Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications—*Data Mining*

## General Terms

Algorithms, Experimentation

## Keywords

Relation Mining, Time-constrained Probabilistic Factor Graph, Dynamic Collaboration Network

## 1. INTRODUCTION

Nowadays, people and entities are connected in all kinds of information networks. With the rapid growth of online networking applications, e.g. Facebook, Twitter and MySpace, it is recognized

that there is abundant information contained in either online or real-world networks. By analyzing the linked data people are able to extract insightful knowledge such as community structure and authoritative sources. Traditionally, the semantic knowledge contained in the links is rarely distinguished further than the known relations attached to them. Given the trend that more and more applications that have done interesting things with data add semantic meaning to the Web, like Google Search Options and Rich Snippets, Feedly, etc., it remains a question how to obtain finer semantic knowledge from plain links. In principle, links can be refined, classified and distinguished though they are mingled in the firsthand data. For instance, in a forum where replied messages can be seen as links connecting different users within the same discussion board, these links can carry very different meanings, e.g. supportive/friends or opposite/opponents. To better understand the network composition it is important to differentiate them.

Due to the large scale of real networks, it is usually difficult, if not infeasible, to distinguish the semantic meaning of every link by labeling. On the other hand, people have assumed and verified there is correlation from link information to the semantic content on Web and many other information networks[11]. Such assumptions provide the possibility to discover the latent difference behind the links by analyzing the network itself. Now we explore it to see whether we are able to excavate the deep semantic meaning of relations under a network with undistinguished nodes and links, based on a small set of intuitive assumptions.

In this work, we study the problem mining the advisor-advisee relationship between coauthors from collaboration network. There are some interesting problems in academic network such as how the research community is formed by each individual researcher. Identifying advisor-advisee relationship can give us better intuition to answer these questions, as it will provides additional semantics of the links other than the unique relation between cooperators. For example, we can position each person in a chronological axis in a correct order, and sketch the whole community in a clear view. As this advising relation is of interest and also convenient to evaluate, we choose it as a case study of the general problem of relation mining on networks.

Many projects have been set up to maintain advisor information for various research fields. However, all of these projects rely on manually collecting the academic genealogy data which makes them quite costly. Therefore, we need to develop a general analyzing technique in order to automatically mine the relations from the network data. Given a collaboration network of researchers, their relations are hidden in the collaboration records and characterized by the development of each researcher and the change of their social role. Regardless of means, it is difficult to differentiate the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'10, July 25–28, 2010, Washington, DC.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

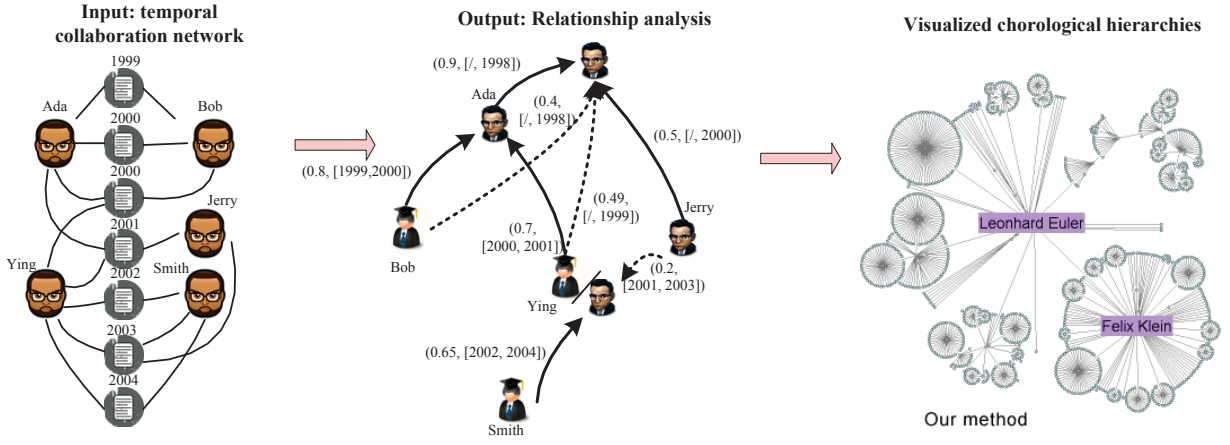


Figure 1: Example of advising relationship analysis on the co-author network.

social role from the static collaboration network. Thus we consider the temporal information and build a unified model for the dynamic collaboration network. Our methodology is expected to apply for other dynamic network and other relations related to time.

There is a vast amount of collaboration information network available online. In Figure 1, the left figure shows an example of coauthor network. Our analysis takes the co-author network as input and outputs the potential advisors of each author and ranks them according to the probability, as shown in the middle figure. For each potential advising relationship, e.g., the relationship between Frank and Carol, the analysis generates an edges directed from the advisee to the advisor, each associated with a vector  $(r, [st, ed])$ , composed of the ranking score  $r$  and the advising time interval  $[st, ed]$ . If one's advisor is not in the network, the parent is a virtual node, which is also the root of the whole network. With the discovered advising relation, we can study other problems such as research community detection and evolution analysis. The right figure gives an example of visualized chorological hierarchies.

The problem we studied is quite different from existing works on relation mining. Our work analyzes homogeneous links rather than text or labeled annotations, and poses a set of challenges.

- *Latent relation.* The advisor-advisee relation is completely hidden in the collaboration links. There is no extra annotation indicating who is one's advisor among numerous collaborators.
- *Time-dependent.* Social role like advisor or advisee is highly time-dependent. One could turn from an advisee to an advisor but there is no clear sign when this transition happens.
- *Scalability.* To find one's advisor it is not sufficient to only consider the information of his coauthors. The network as a whole is correlated and the search space is exponential in size. It is important to develop a method that can scale well to real large data sets.

In this paper, we formulate the problem of advising relationship mining as a probabilistic ranking problem, and propose a time-constrained probabilistic factor graph (TPFG) model to model the dynamic collaboration network. An efficient algorithm to optimize the joint probability and obtain ranking score is designed as a process of message propagation on the network.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 formally formulates the problem. Section 4

explains the proposed approach. Section 5 presents experimental results that validate the computational efficiency and efficacy of our methodology. Finally, Section 6 concludes and discusses future work.

## 2. RELATED WORK

Existing works of relation mining mainly employs text mining and language processing technique on text data and structured data including web pages, user profiles and corpus of literatures. Semantic Role Labeling is a broadly employed text mining technique, as it allows for the addition of structured semantic information to plain text [8]. [12] applies Natural Language Processing to extract protein-protein relations etc. from rich-annotated corpus in biomedical domain. [18] analyzes relation of elements in XML on the Web. And [5] proposes a general framework for syntax-based relation mining and achieves high accuracy by experimenting with Support Vector Machine as a supervised approach. To the knowledge of the authors there is no previous work mining semantic relations solely from a network without annotation text based on assumptions about the correlation between the observed links and the latent roles of the nodes.

By means of link mining technique, people can compute importance of a node, and relevance of neighboring nodes, e.g. using PageRank[2]. Since then there have been good examples showing how the network has the power to clean, fuse and reveal the knowledge hidden in it, such as *Object Distinction*[16] and *Veracity*[17]. The former distinguishes objects with identical names by analyzing their heterogeneous linkages, while the latter studies how to find true facts from a large amount of conflicting information provided by various web sites. Furthermore, we can do ranking and clustering on heterogeneous network based on the link information, e.g. using NetClus[13]. Recognizing the power of links, we develop our approach to mine the relation by modeling the network.

To evaluate the discovered advisor-advisee relations we can compare with known graduation records maintained by some online projects. These include the Mathematics Genealogy Project[4], the Computer Engineering Academic Genealogy, the AI Genealogy Project and the Software Engineering Academic Genealogy. [15] proposes an approach based on classification to classify the relations according to some local features on each pair of coauthors but the parameters are manually tuned. We develop their method into a supervised learning process and compare it with our probabilistic-model-based approach.

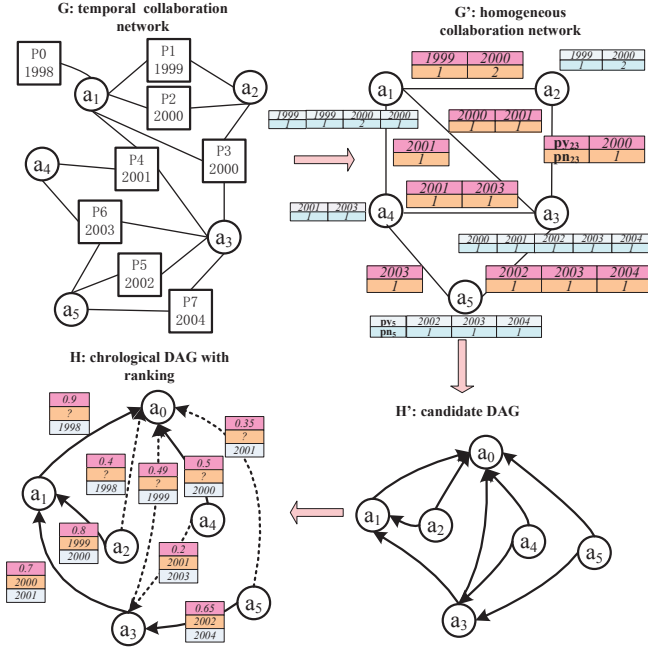


Figure 2: Example of graph transformation.

### 3. PROBLEM FORMULATION

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

In general, the input is a time-dependent collaboration networks  $\{G\} = \{(V = V^p \cup V^a, E)\}$ , where  $V^p = \{p_1, \dots, p_n\}$  is the set of publications, with  $p_i$  published in time  $t_i$ ,  $V^a = \{a_1, \dots, a_{n_a}\}$  is the set of authors, and  $E$  is the set of edges. Each edge  $e_{ij} \in E$  associates the paper  $p_i$  and the author  $a_j$ , meaning  $a_j$  is one author of  $p_i$ .

The original heterogeneous network can be transformed into a homogeneous network containing only authors immediately. Let  $G' = (V', E', \{\mathbf{py}_{ij}\}_{e_{ij} \in E'}, \{\mathbf{pn}_{ij}\}_{e_{ij} \in E'})$ , where  $V' = \{a_0, \dots, a_{n_a}\}$  is the set of authors including the virtual one, and  $E'$  is the collaboration records. Each edge  $e'_{ij} = (i, j) \in E'$  connects two coauthors if and only if they have publication together, and there are two vectors associated with the edge,  $\mathbf{py}_{ij}$  and  $\mathbf{pn}_{ij}$ . They are of equivalent length, indicating the time they have publications and the number of coauthored papers they have in that time. For example,  $\mathbf{py}_{ij} = (1999, 2000, 2001)$ ,  $\mathbf{pn}_{ij} = (2, 3, 4)$  indicates that the author  $a_i$  and  $a_j$  have coauthored 2, 3 and 4 papers in three years 1999, 2000 and 2001 respectively. Further more, we denote the number of papers  $a_i$  have published in each year with two similar vectors  $\mathbf{py}_i$  and  $\mathbf{pn}_i$ . They can be derived from  $\mathbf{py}_{ij}$  and  $\mathbf{pn}_{ij}$ .

We assume an author  $a_i$ 's advisor is  $a_{y_i}$ , in which the index  $y_i$  is a hidden variable. If  $a_i$  is advised by  $a_j$ , we use  $[st_{ij}, ed_{ij}]$  to represent the time interval the advising relation lasts. Especially we let  $st_i = st_{iy_i}$  and  $ed_i = ed_{iy_i}$ . And if  $a_i$  is not advised by anybody in the database, we let  $y_i = 0$  to direct  $a_i$ 's advisor to a virtual node  $a_0$ .

In this setting, to find the advisor and advising time for each author  $a_i$ , we need to decide  $y_i$  as well as  $st_{iy_i}, ed_{iy_i}$ . In reality, this problem is more complicated than finding one doctoral advisor from all the coauthors of a given researcher because: first, one could have multiple advisors like master advisors, PhD co-advisors, post-doctoral advisors; second, one's advisor could be missing in the data. Therefore, instead of using boolean model, we adopt a probabilistic model to rank the likelihood of several potential ad-

visors for each author. Formally, the notation  $r_{ij}$  is used for the ranking score of  $a_j$  being  $a_i$ 's advisor. To reduce the number of authors that needed to be ranked, it is beneficial to keep only those potential pairs of advisor-advisee. We can construct a sub graph  $H' \subset G'$  by removing some edges from  $G'$  and make the remaining edges directed. Thus  $H' = (V', E'_s)$  and  $E'_s \subset E'$ . Later we will show that it is possible to extract a directed acyclic graph (DAG)  $H'$  from  $G'$ . In  $H'$ , the index set of potential advisors of a given author  $\beta$  is denoted  $Y_\beta$ , e.g.  $Y_3 = \{0, 1\}$ . Correspondingly, the index set of potential advisees is denoted  $Y_i^{-1}$ .

Then the task becomes to decide  $r_{ij}, st_{ij}, ed_{ij}$  for every possible advising pair  $(i, j) \in E'_s$ . So the output is the DAG  $H = (V', E'_s, \{(r_{ij}, st_{ij}, ed_{ij})\}_{(i,j) \in E'_s})$ . After the chronological DAG  $H$  is constructed, the ranking score can be used to predict whether there is an advisor-advisee relationship between every pair of coauthors  $(a_i, a_j)$ . A simple way to do the prediction is to fetch top  $k$  potential advisors of  $a_i$  and check whether  $a_j$  is one of them while  $r_{ij} > r_{i0}$  or  $r_{ij} > \theta$ , where  $\theta$  is a threshold such as 0.5. We use  $P@k, \theta$  to denote this method. It is predictable that large  $k$  and large  $\theta$  leads to better recall and worse precision. How to choose  $k$  and  $\theta$  could be a tricky problem. So we allow the input contains some training data so as to decide the parameters. If no training data are provided, we can simply use some empirical values.

The transformation process from one graph to another is illustrated in Figure 2.

### 4. APPROACH

We have formalized the problem and defined the transformation process to solve the problem. In this section, We first make basic assumptions as the prerequisite of our approach, and proposes a two-stage framework. Then the approach for each stage is presented. The main idea is to use a time-constrained probabilistic factor graph model to decompose the joint probability of the unknown advisor of every author. The dynamic change of the collaboration ratio between authors, as well as the time constraint on the relationship due to the hidden social role, is captured via factor functions, which form the basic components of the factor graph model. By maximizing the joint probability of the factor graph we can infer the relationship and compute ranking score for each relation edge on the candidate graph. To compute it we can use the general algorithm sum-product and JunctionTree. However, the standard algorithm suffers from the problem of low efficiency. we design a message passing algorithm on the candidate graph to approximate the computation and improve the efficiency greatly.

#### 4.1 Assumptions and Framework

We have the following assumptions based on common sense.

ASSUMPTION 1.

$$\forall 1 \leq x \leq n_a, ed_{y_x} < st_x < ed_x \quad (1)$$

This simple formula reflects the following fact for general consideration of advising relationship. At each time  $t$  during the publication history of a node  $u$ ,  $u$  is either being advised or not being advised. Once  $u$  starts to advise another node, it will never be advised again.  $u$  cannot advise  $v$  at the year  $t_1$  if  $u$  is advised by any node  $p$  at the year  $t_1$ . If  $u$  advises  $v$ , the time  $v$  is advised by  $u$  is a continuous interval from  $t_1$  to  $t_2$ ,  $t_1 < t_2$ . As a result of Assumption 1, we need to infer the advisors of all the nodes in the network together, rather than consider them separately. In Section ??, we will use this assumption in our model.

ASSUMPTION 2.

$$\forall 1 \leq x \leq n_a, py_{y_x}^1 < py_x^1 \quad (2)$$

That means for a given pair of advisor and advisee, the advisor always has a longer publication history than the advisee.  $py_x^1$  represents the first component of vector  $\mathbf{py}_x$ . Assumption 2 determines that all the authors in the network have a strict order defined by the possible advising relationship. It is easy to verify the property of irreflexivity, asymmetry and transitivity. Due to the order, the candidate graph  $H'$  is assured to be a DAG. We will use this assumption in the filtering process.

To measure the likelihood of the potential relationship according to people's publication history, we need some prior knowledge about their correlation as additional assumptions. They will be posed in Section 4.2. Now we propose a two-stage framework solution for the advisor-advisee relationship mining problem. In stage 1, we preprocess the heterogeneous collaboration network to generate the candidate graph  $H'$ . That includes the transformation from  $G$  to homogeneous network  $G'$ , the construction from  $G'$  to  $H'$  and the estimate of the local likelihood on each edge of  $H'$ . In stage 2, these potential relations are further modeled with a probabilistic model. Local likelihood and time constraints are combined in the global joint probability of all the hidden variables. The joint probability is maximized and the ranking score of all the potential relations is computed together. The construction of  $H$  is finished in this stage.

## 4.2 Stage 1: Preprocessing

The purpose to do preprocessing is to generate the candidate graph  $H'$ , reducing the search space while keeping the real advisor is not excluded from the candidate pool in most cases. First we need to generate according to the collaboration information a homogeneous author network  $G'$  by processing the papers in the network one by one. For each paper  $p_i \in V^p$ , we construct an edge between every pair of its authors and update the vectors  $\mathbf{py}$  and  $\mathbf{pn}$ . The complexity of this process is  $O(\sum_{p_i \in V^p} d_i^2)$ , where  $d_i$  is the degree of  $p_i$  in  $G$ .

Then we launch a filtering process to remove unlikely relations of advisor-advisee. For each edge  $e_{ij}$  on  $G'$ ,  $a_i$  and  $a_j$  has collaboration. To decide whether  $a_j$  is  $a_i$ 's potential advisor, we check the following conditions. First, Assumption 2 is checked. Only if  $a_j$  started to publish earlier than  $a_i$ , we consider the possibility. Second, we apply some heuristic rules, which are based on the prior intuitive knowledge about advisor-advisee relations. It is unknown how appropriate they are before we test the results. Thus we list the rules here, and will test them in the experiment part.

First we introduce two measures for the coauthored publications between any pair of cooperators, Kulczynski and IR. They are defined as

$$kulc_{ij}^t = \frac{\sum_{py_{ij}^k \leq t} pn_{ij}^k}{2} \left( \frac{1}{\sum_{py_i^k \leq t} pn_i^k} + \frac{1}{\sum_{py_j^k \leq t} pn_j^k} \right) \quad (3)$$

$$IR_{ij}^t = \frac{\sum_{py_j^k \leq t} pn_j^k - \sum_{py_i^k \leq t} pn_i^k}{\sum_{py_i^k \leq t} pn_i^k + \sum_{py_j^k \leq t} pn_j^k - \sum_{py_{ij}^k \leq t} pn_{ij}^k} \quad (4)$$

The Kulczynski measure reflects the correlation of the two authors' publication. [14] shows that there usually exists high correlation between the total publications of advisors and advisee. Here we further incorporate the time factor, to calculate the measure year by year, and check whether there is a increase in the sequence  $\{kulc_{ij}^t\}_t$ . For IR we calculate the sequences in the same way.

IR [14] is used to measure the imbalance of the occurrence of  $a_j$  given  $a_i$  and the occurrence of  $a_i$  given  $a_j$ . The intuition is that the advisor has more publication records than the advisee during the advising time. Then we have the following rules.

- R1: If  $IR_{ij}^t < 0$  in the sequence  $\{IR_{ij}^t\}_t$  during the cooperation period of  $a_i$  and  $a_j$ ,
- R2: If there is no increase in the sequence  $\{kulc_{ij}^t\}_t$  during the cooperation period,
- R3: If the cooperation period of  $a_i$  and  $a_j$  lasts only for one year,
- R4: If  $py_j^1 + 2 > py_{ij}^1$ ,

then  $a_j$  is not considered to be  $a_i$ 's advisor. When the pair of authors passes the test of selected rules from them, we construct a directed edge from  $a_i$  to  $a_j$  in  $H'$ . In addition, we estimate the starting time and ending time of the advising, as well as the local likelihood of  $a_j$  being  $a_i$ 's advisor  $l_{ij}$ . For the estimation we also have various methods. The starting time  $st_{ij}$  is estimated as the time they started to collaborate, while the ending time  $ed_{ij}$  can be estimated as either the time point when the Kulczynski measure starts to decrease, or the year making the largest difference between the Kulczynski measure before and after it. We refer to the two method as YEAR1 and YEAR2. And we refer to YEAR as taking the earlier time of the two years estimated by them. After we estimate  $st_{ij}$  and  $ed_{ij}$ , we calculate the average of Kulczynski and IR measure during that period, and use one of them or average the two as the local likelihood, like the following.

$$l_{ij} = \frac{\sum_{st_{ij} \leq t \leq ed_{ij}} (kulc_{ij}^t + IR_{ij}^t)}{2(ed_{ij} - st_{ij} + 1)} \quad (5)$$

The complexity of processing each edge is  $O(T)$ , if we assume the oldest paper and the newest one differs  $T$  in their publication time. The total complexity to transform  $G'$  to  $H'$  is  $O(MT)$ , where  $M$  is the number of edges in  $G'$ .

## 4.3 Stage 2: TPGF Model

From the candidate graph  $H'$  we know the potential advisors of each author and the likelihood based on local information. By modeling the network as a whole we can incorporate both structure information and temporal constraint and better analyze the relation of each individual links. Now we formally define the proposed TPGF model.

For each node  $a_i$ , there are 3 variables to decide,  $y_i, st_i, ed_i$ . Suppose we have already had a local feature function  $g(y_i, st_i, ed_i)$  defined on the three variables of any given node. To model the joint probability of all the variables in the network, we define it as the product of all local feature functions.

$$P(\{y_i, st_i, ed_i\}_{a_i \in V^a}) = \frac{1}{Z} \prod_{a_i \in V^a} g(y_i, st_i, ed_i) \quad (6)$$

with

$$\forall a_i \in V^a, ed_{y_i} < st_i < ed_i \quad (7)$$

where  $\frac{1}{Z}$  is the normalizing factor of the joint probability

Equation 7 is the constraint according to Assumption 1. To find most probable values of all the hidden variables, we need to maximize the joint probability of all of them. To estimate the approximate size of entire search space, assume each author has  $C$  candidates and the advising time can vary in a range of  $T$ , then the

combination of all the variables has exponential size  $(CT^2)^{n_a}$ . It is intractable to do exhaustive search. We make a first simplification as follows. Suppose  $a_i$  and his advisor  $y_i$  is given. Instead of letting  $st_i$  and  $ed_i$  vary, we fix them by optimizing local function  $g(y_i, st_i, ed_i)$ , i.e.,

$$\{st_i, ed_i\} = \arg \max_{st_i < ed_i} g(y_i, st_i, ed_i) \quad (8)$$

In this way  $st_i$  and  $ed_i$  are tied to the value of  $y_i$ . Once  $y_i$  is decided, they are derived correspondingly. We can pre-compute the best advising time as  $st_{ij}$  and  $ed_{ij}$  for each  $y_i = j$ . Now only  $\{y_i\}$  are variables to optimize. If we embed the constraint Eq. 7 into the feature function, the objective function becomes

$$P(y_1, \dots, y_{n_a}) = \frac{1}{Z} \prod_{i=1}^{n_a} f_i(y_i | \{y_x | x \in Y_i^{-1}\}) \quad (9)$$

with

$$f_i(y_i = j | \{y_x | x \in Y_i^{-1}\}) = g(y_i, st_{ij}, ed_{ij}) \prod_{x \in Y_i^{-1}} I(y_x \neq i \vee ed_{ij} < st_{xi}) \quad (10)$$

where

$$I(y_x \neq i \vee ed_{ij} < st_{xi}) = \begin{cases} 1 & y_x \neq i \vee ed_{ij} < st_{xi} \\ 0 & y_x = i \wedge ed_{ij} \geq st_{xi} \end{cases} \quad (11)$$

is the identity function. In this way the time constraints Eq. 7 for all hidden variables are decomposed to many local identity function. Now we only need to optimize Eq. 9. Furthermore, to obtain the rank score of each advising relationship, e.g.  $a_j \xrightarrow{\text{advise}} a_i$ , we can compute the conditional maximal probability

$$r_{ij} = \max P(y_1, \dots, y_{n_a} | y_i = j) \quad (12)$$

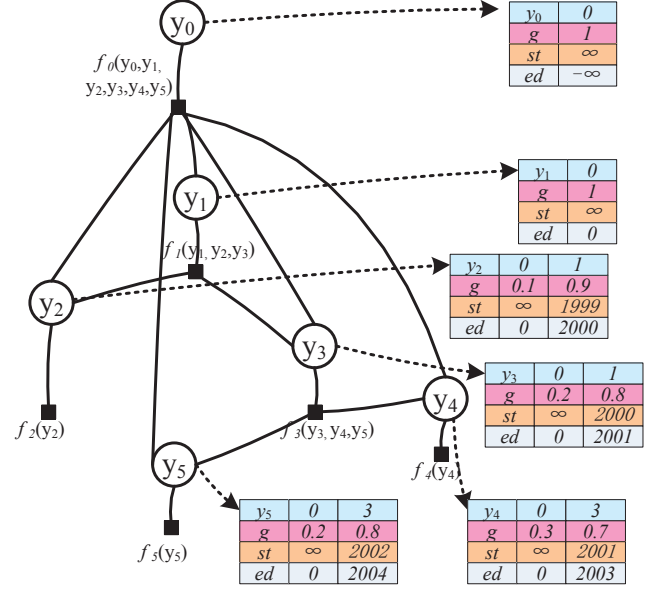
The search space size now becomes  $C^{n_a}$ , reduced but still exponential. Since we have decomposed the dependency of the variables, we can use a factor graph model to accomplish efficient computation.

Figure 3 shows a simple TPFPG corresponding to the example we have been using so far. The graph is composed of two kinds of nodes: variable nodes and function nodes. Variable nodes map to the hidden variables  $\{y_i\}_{i=0}^{n_a}$ . Each variable node corresponds to a function node  $f_i(y_i | \{y_x | x \in Y_i^{-1}\})$ . All of the edges are of one kind, connecting a variable node with a function node. There is one edge between one variable node  $y_x$  and a function node  $f_i(\cdot)$  if and only if  $f_i(\cdot)$  depends on  $y_x$ . In our case, it is equivalent with  $x = i$  or  $x \in Y_i^{-1}$  (a.k.a.  $i \in Y_x$ ). The factor graph reflects the dependency of the variables. A set of variables are correlated if they are neighbors of the same function node, e.g.  $y_1, y_2, y_3$  with  $f_1(y_1, y_2, y_3)$ . There is additional information stored in each variable node, as shown as the tables in Figure 3.  $y_i$  can take different values from  $Y_i$ , and the corresponding  $st_i, ed_i$  and  $g(y_i, st_i, ed_i)$  are pre-computed in stage 1. Here we take  $l_{ij}$  as  $g(y_i, st_{ij}, ed_{ij})$  when  $y_i = j$ .

Theoretically, one can incorporate any types of features into the TPFPG model. For example, one can even define some constraints like: One advisor has no more than 5 advisees whom he starts to advise at the same year. For different kind of relationship, the constraint can vary according to primary assumptions.

#### 4.4 Model Learning

To maximize the objective function and compute the ranking score along each edge in candidate graph  $H'$ , we need to infer the



**Figure 3: Graphical representation of the time-constrained probabilistic factor graph.**  $\{y_0, \dots, y_5\}$  are hidden variables defined on all nodes;  $f_i(\cdot)$  represents a factor function defined on a hidden variable and its potential advisee sets as neighbors.

marginal maximal joint probability on TPFPG, according to Eq. 12. We first introduce the algorithm for general factor graph, discuss its deficiency and then propose our algorithm.

**sum-product + junction tree** There is a general algorithm called *sum-product* [9] to compute marginal function on a factor graph based on message passing. It performs exact inference on a factor graph without cycles. In sum-product algorithm, marginal functions of a single variable, a.k.a. messages are passed between neighboring variable node and function node. To compute the marginal maximal probability, we need to change sum-product to max-sum with a logarithmic transformation of the function value. If TPFPG is tree-structured factor graph, the message passing rule will be:

$$m_{y_i \rightarrow f_j(\cdot)}(y_i) = \sum_{j' \in Y_i \cup \{i\}, j' \neq j} m_{f_{j'}(\cdot) \rightarrow y_i}(y_i) \quad (13)$$

$$m_{f_j(\cdot) \rightarrow y_i}(y_i) = \max_{\sim \{y_i\}} (\log f_j(y_i, \{y_{i'}\}) + \sum_{i' \in Y_j^{-1} \cup \{j\}, i' \neq i} m_{y_{i'} \rightarrow f_j(\cdot)}(y_{i'})) \quad (14)$$

where  $j' \in Y_i \cup \{i\}, j' \neq j$  represents  $f_{j'}(\cdot)$  is a neighbor node of variable  $y_i$  on the factor graph except factor  $f_j(\cdot)$ ,  $\sim \{y_i\}$  represents all variables in  $Y = \{y_1, \dots, y_{n_a}\}$  except  $y_i$ .

Unfortunately TPFPG contains cycles. This algorithm cannot be applied directly. One solution to generalize it is a procedure known as *junction tree algorithm* [1] 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. The process to construct a junction tree alone consumes a lot in both time and space. In practice we found it fails to finish for 3000 variables, not to mention our TPFPG has the scale of more than 600,000 variables.

To reduce the computational cost, we can do approximate inference instead of exact inference. One approach *loopy belief propa-*



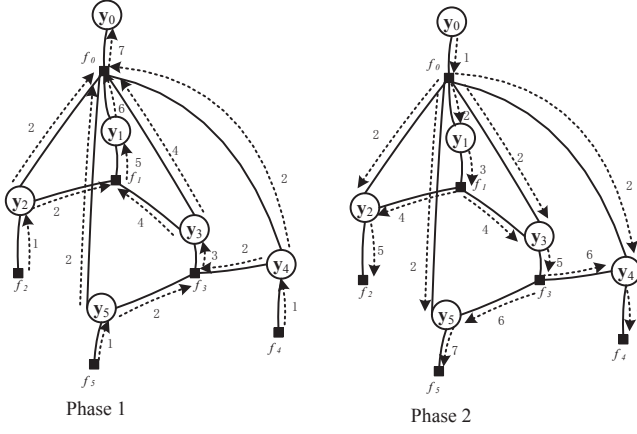


Figure 4: The 2-phase message passing schema.

gation[6] simply applies the sum-product algorithm in cycle-containing graph. To avoid repetitive information flow for multiple times through the graph, we design a special message passing schedule and the following algorithm according to the special property of TPGF.

**New TPGF Inference Algorithm** The original sum-product or max-sum algorithm meet with difficulty since it requires that each node need wait for all-but-one message to arrive, thus in TPGF some nodes will be waiting forever due to the existence of cycles. To overcome this problem, we arrange the message passing in the a mode based on the strict order determined by  $H'$ . Each node  $a_i$  has a descendants set  $Y_i^{-1}$  and an ascendants set  $Y_i$ .

As shown in Figure 4, the message is passed in a two-phase schema. In the first phase, messages are passed from advisees to advisors, and in the second, messages are passed back from advisors to advisees. Formally there are two kinds of messages in the first phase:  $m_{f_i() \rightarrow y_i}, m_{y_i \rightarrow f_j()}$  where  $j \in Y_i$ . The message from  $f_i()$  to  $y_i$  is generated and sent only when all the messages from its descendants have arrived. And  $y_i$  immediately send it to all its ascendants  $f_j(), j \in Y_i$ . In phase two, there are also two kinds of messages:  $m_{y_i \rightarrow f_i()}, m_{f_j() \rightarrow y_i}, j \in Y_i$ , every of which are along the reverse direction on the edge as in phase 1. The order of message passed is illustrated by the number on each edge in Figure 4. The messages are calculated according to Eq. 14 and 13.

This algorithm still has redundant computation. We further simplify the message propagation by eliminating the function nodes and the internal messages between a function node and a variable node, and transforming it to a message passing problem on the homogeneous graph  $H'$ . Messages are propagated between authors, and the messages can be stored with each author in two vectors, one sent and one received. In this way both time and space are saved. The rules are formally defined as follows. (Derivation of the update rule is omitted.)

$$\text{sent}_{ij} = \log l_{ij} + \sum_{k \in Y_i^{-1}} \max_{st_{kx} > ed_{ij} \text{ or } x \neq i} \text{sent}_{kx} \quad (15)$$

$$\begin{aligned} \text{recv}_{ij} = & \max_{j' \in Y_j, ed_{jj'} < st_{ij}} (\text{recv}_{jj'} + \log l_{jj'} + \\ & + \sum_{k \in Y_j^{-1}, k \neq i} \max_{x \in Y_k, st_{kx} > ed_{jj'} \text{ or } x \neq j} \text{sent}_{kx}) \\ & + \sum_{x \in Y_i, x \neq j} \max_{j' \in Y_x} (\text{recv}_{xj'} + \\ & + \sum_{k \in Y_x^{-1}, k \neq i} \max_{x' \in Y_k, st_{kx'} > ed_{xj'} \text{ or } x' \neq x} \text{sent}_{kx'}) \end{aligned} \quad (16)$$

$$r_{ij} = \exp(\text{sent}_{ij} + \text{recv}_{ij}) \quad (17)$$

**Input:**  $H' = (V', E'_s, \{st_{ij}, ed_{ij}, l_{ij}\}_{(i,j) \in E'_s})$   
**Output:**  $H = (V', E'_s, \{(r_{ij}, st_{ij}, ed_{ij})\}_{(i,j) \in E'_s})$   
 Calculate the logarithm of local feature function  $l_{ij}$ ;  
 Initialize all  $\text{sent}_{ij}$  as  $\log l_{ij}$ ;  
 Initialize a counter for each node  $count_i \leftarrow |Y_i^{-1}|$ ;  
 Initialize a stack-queue  $Q$ , enqueue all the nodes  $x$  s.t.  $count_x = 0$ ;  
**repeat**  
    $i \leftarrow$  the head of  $Q$ ;  
   Increment the head pointer of  $Q$  by 1;  
   **foreach** edge  $(i, j), j \in Y_i$  **do**  
     Update  $\text{sent}_{ij}$  according to Eq. 15;  
      $count_j \leftarrow count_j - 1$ ;  
     **if**  $count_j == 0$  **then**  
       enqueue  $j$ ;  
     **end**  
   **end**  
**until** the head of  $Q$  is 0;  
 Treat  $Q$  as a stack, let  $top$  points to the tail; **repeat**  
   Pop the top element of  $Q$  to  $j$ ; **if**  $j == 0$  **then**  
      $\text{recv}_{j0} \leftarrow 0$   
   **end**  
   **else**  
     **foreach**  $j' \in Y_j$  **do**  
       Collect  $\text{recv}_{jj'}$  and  $\text{sent}_{jj'}$  to compute  $r_{jj'}$  according to Eq. 17 and prepare to compute  $\text{recv}_{ij}$ ;  
     **end**  
   **end**  
   **foreach**  $i \in Y_j^{-1}$  **do**  
     Compute  $\text{recv}_{ij}$  according to Eq. 16;  
   **end**  
**until**  $Q$  is not empty;  
 Generate  $H = (V', E'_s, \{(r_{ij}, st_{ij}, ed_{ij})\}_{(i,j) \in E'_s})$

Algorithm 1: The improved TPGF inference algorithm.

In the new algorithm, the message propagation can be done by using a stack-queue. In phase 1, each node will enter the queue once and the vector  $\text{sent}_i$  for them is computed one by one. In phase 2, we scan the queue from the tail back to the head, i.e. treat it as a stack, and compute  $\text{recv}_i$ . Then we can normalize the results and collect them to get the ranking score. By using  $O(|E'_s|)$  space, the running time of the algorithm can be reduced to  $O(\sum_{i=1}^{n_a} d_i d'_i)$ , where  $d_i$  and  $d'_i$  is the in-degree and out-degree of each node  $a_i$  on graph  $H'$ . Since  $H'$  is sufficiently sparse, the maximal degree of the node can be seen as constant  $C$ . In this case the complexity further reduce to  $O(n_a)$ .

## 5. EXPERIMENTAL RESULTS

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

### 5.1 Experiment Setup

**Data Sets** We perform experiments on several real-world data sets. We use the DBLP Computer Science Bibliography Database maintained by Michael Ley as the dynamic collaboration data set  $G$  to infer the advisor-advisee. The data set consists of 654,628 authors and 1,076,946 publications with the year of each publication provided. The publication records cross over the time frame from 1970 to 2008. To test the accuracy of the discovered advisor-advisee relationship, we adopt 2 data sets. One is manually labeled by looking into the home page of the advisors. The other is crawled from Mathematics Genealogy project<sup>1</sup>. We refer to them as MAN and MathGP respectively. We further separate MAN into three sub data sets: Teacher, PhD and Colleague. Teacher contains hundreds of miscellaneous advisor-advisee pairs, while PhD only contains 100 graduated PhD pairing with their advisor. Colleague contains hun-

<sup>1</sup><http://www.genealogy.math.ndsu.nodak.edu/>

dreds of colleague pairs which are negative samples for advisor-advisee relationship.

**Method** We compare our improved TPGF inference algorithm (will be referred as TPGF for short) with the following approaches: sum-product+junction tree(JuncT); local maximal (LOCAL); and SVM. The first two generate the same form of ranking as output as TPGF does. The difference is that JuncT computes the exact joint probability as the ranking score while LOCAL adopts the local likelihood obtained by preprocessing. They do not need training data while SVM needs some labeled pairs, both positive and negative, as training data. We also compare with a baseline method MCP: for each author, from all the cooperators that satisfy Assumption 2, choose the one with Most Coauthored Papers with him.

#### Evaluation Aspects

To quantitatively evaluate our method, we consider three performance issues:

- **Accuracy.** We use several accuracy test to demonstrate how effective our method can learn the latent advisor-advisee relations.
- **Running time.** It is the execution elapsed time of the computation. This determines how efficient our method is.
- **Applications.** We apply the identified relationship to some application. This will demonstrate how the constructed relations and hierarchies can benefit other information network-ing applications.

The filtering process is implemented using MATLAB 2009a and all experiments with it are performed on a Server running Windows XP with two Dual-Core Intel Pentium 4 processors (3.0 GHz) and 1GB memory. The JuncT algorithm is implemented using the package MALLET[10], which is an integrated collection of Java code for machine learning. We implement TPGF with Microsoft Visual C++ 2008. And we use LIBSVM[3] to perform SVM training and prediction.

## 5.2 Accuracy

We conduct a series of experiments to explore the capability of TPGF algorithm in mining advisor-advisee relationship in the network. First, as we mentioned in Section 4.1 and Section 4.2, different assumptions about advising relationship can be tested to see which of them are the best combination to reflect the reality. Second, we extract small fraction of the whole DBLP network and feed as the original input  $G$  to TPGF, to prove that the power of network boosts the estimation of joint probability. Finally, we compare our unsupervised approach with other approach based on classification that requires labeled data for training. We also tested whether TPGF can be further improved by utilizing training data.

### 5.2.1 rules validation

We try to use the rules one by one gradually to construct corresponding candidate graph  $H$ , compute the ranking score with our algorithm, and compare the accuracy of advising relation prediction on some labeled data.

The accuracy is compared through ROC curve. For each pair in the tested data, we retrieve the ranking score from the output of the algorithm TPGF. Then we sort these ranking score in a descendant order, and draw the ROC curve for them.

From Figure 5 we can see that R3 have the highest suitability on the tested data. We can further refine the rules in this way to determine such as the local likelihood evaluation measure and the graduation year estimation method, as shown in Figure 6. It is also observed that TPGF is not sensitive to some of the rules. In general,

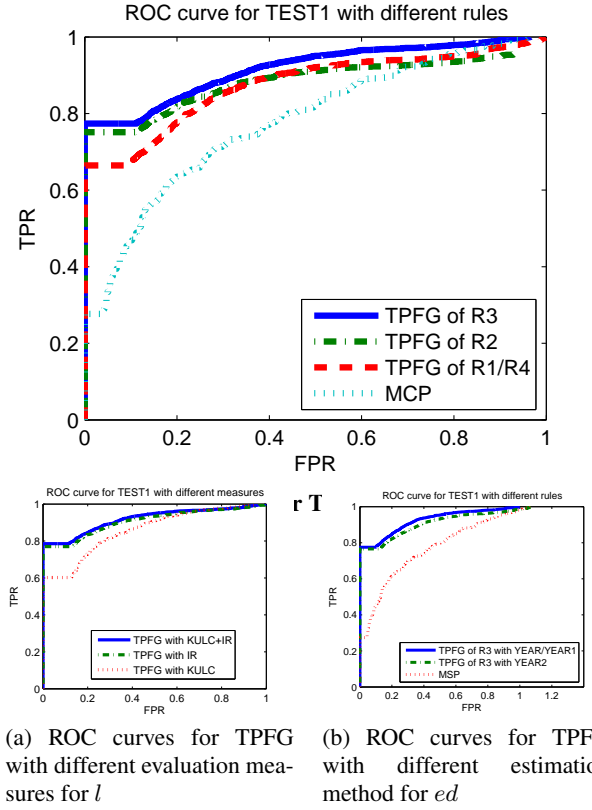


Figure 6: rule refinement.

by applying a small set of rules our method can extract meaningful knowledge, while which rules to select is not a problem to prevent its efficacy.

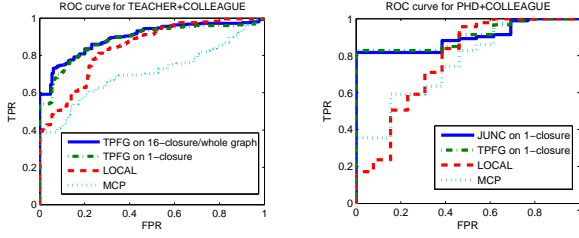
### 5.2.2 boosts by the network

With the help of DFS with a bounded maximal depth  $d$  from the given set of nodes, which we denote DFS- $d$  for, we can obtain closures with controlled depth for a given set of pairs of authors to test. When  $d$  increases, the subnetwork grows larger. The growth will ends when the subnetwork is already the complete closure, i.e. the maximal connected subgraph of  $H$  containing the given set. We run TPGF for these closures and draw the ROC curve for them. From Figure 7(a) we see that with different scales of closures provided, TPGF achieves better accuracy when the depth increases, and they all outperform the LOCAL method. And on the complete closure TPGF achieves the same accuracy as on the whole network since the disconnected components will not affect each other.

On these various scaled subnetworks, TPGF achieves different level of approximation to the optimal global joint probability on the whole network. To compare it with the exact maximal joint probability, we construct 1-closure which contains 1310 nodes for a small data set because JuncT fails on larger network due to the high requirement for the space. From Figure 7(b) we find that in the small graph TPGF approximates well with the exact inference algorithm JuncT.

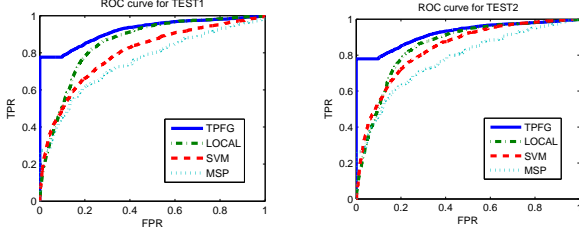
### 5.2.3 supervised approach

SVM is a special form of Logistic Regression classification method. If we treat an advisor-advisee pair as a positive example and a non advisor-advisee pair of coauthors as a negative example, we can reduce the advisor mining to a classification problem on the ordered pairs  $(a_i, a_j)$ . To make sure the classifier make use of no less information than our probabilistic model, we add Kulczynski and IR



(a) ROC curves for TPFPG on different closures (b) ROC curves for Junc and TPFPG on smaller graphs

Figure 7: Test for various scales of subnetworks.



(a) ROC curves for LOCAL, TPFPG and SVM on TEST1 (b) ROC curves for LOCAL, TPFPG and SVM on TEST2

Figure 8: Comparison with training based method.

measure as features beyond what were used in [15]. Direct application of SVM only shows the result of judging whether a given pair is an advisor-advisee pair, and it often finds multiple advisors for a certain author. We examine the probabilistic score for each pair in the test data, and rank them from high to low to draw the ROC curve. We also compare the prediction precision by  $P@k$ .

Although in this paper we restrict our model as a unsupervised learning approach, it can also be modified to achieve supervised learning. For example, we can adjust the parameter  $\theta$  in the  $P@k$ ,  $\theta$  method according to certain criteria such as achieving best information gain on the training data. Then we use the trained method to do prediction on test data. Table 2 shows the improvement by utilizing the training data.

In conclusion, TPFPG can achieve comparable or even better accuracy compared with training based method. When the prediction method according to the ranking obtained TPFPG accepts training data to adjust the parameter, the accuracy will be further improved.

#### 5.2.4 case study

By case study, we also find that TPFPG can discover some interesting relations beyond the "ground truth" from single source. For example, for Ming-Hsuan Yang, we find he is Dan Roth's student according to the latter's homepage. However, the top ranked advi-

Table 1: Accuracy of prediction

data set	MCP	Direct SVM	LOCAL	TPFG
TEST1_POS	63.9%	<b>84.2%</b>	71.5%	72.3%
TEST1_NEG	86.5%	52.1%	85.7%	<b>86.6%</b>
TEST1_ALL	73.5%	64.4%	78.0%	<b>78.8%</b>
TEST2_POS	64.1%	39.7%	71.2%	<b>71.9%</b>
TEST2_NEG	<b>86.3%</b>	83.2%	85.5%	86.3%
TEST2_ALL	68.5%	53.8%	77.7%	<b>78.4%</b>

TRAIN1 = Colleague(491)+PHD(100)  
 TEST1 = Teacher(257)+MathGP(1909)+Colleague(2166)  
 TRAIN2 = Teacher(257)+Colleague(2166)  
 TEST2 = PHD(100)+MathGP(1909)+Colleague(4351)  
 TEST\_POS: recall, \_NEG: precision, \_ALL: F measure

Table 2: Accuracy of prediction

data set	SVM	LOCAL 1	LOCAL 2	TPFG 1	TPFG 2
TEST1	73.4%	77.9%	78.4%	72.9%	83.5%
TEST2	74.6%	78.0%	78.4%	72.8%	83.5%

LOCAL 1,TPFG 1: before training, LOCAL 2,TPFG 2: after

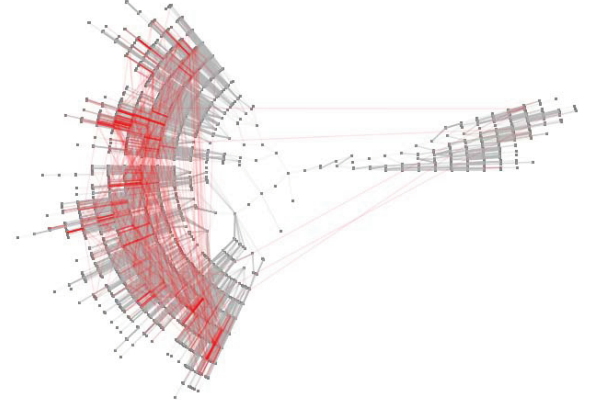


Figure 9: visualized advising relation

sor is Narendra Ahuja. When we refer to Mathematics Genealogy, we find that both are co-advisors of Ming-Hsuan Yang. And Dan Roth is ranked 2nd by TPFPG. The graduation time from them is estimated as 2001 and 2002, separately, which is close to the real time 2000. For another example, Chao Huang is listed as Laxmikant V. Kale's student but by TPFPG the latter is only ranked 4th. Genealogy does not have the record. By google search we find there is a Chao Huang who graduated from Princeton, and the top 3 professors ranked by TPFPG are all from Princeton.

### 5.3 Scalability Performance

Table 3 lists the running time required for those algorithms. The same filtering procedure is used for JuncT, LOCAL and TPFPG. LOCAL and MCP do not do further learning after preprocessing.

Considering the scale of our data set, LOCAL and TPFPG are proved to be scalable. With regard to the total time, TPFPG costs hardly any more than LOCAL or MCP. SVM as a classification approach is essentially different from the other three. Its scalability is relative to the training dataset, and we mainly use it to compare the accuracy.

Table 3: Scalability performance of different methods on two stages.

Data set	MCP	LOCAL	JuncT	TPFG
DBLP ALL	58min	<b>48min</b>	N/A	53min
1-closure of SMALL	N/A	N/A	9min	<b>22s</b>

SMALL=PHD+COLLEAGUE.  $n_a = 1308$ ,  $|E_s| = 3697$  for 1-closure.

### 5.4 Applications[\*\* to complement \*\*]

The relationship analysis can benefit many applications. The visualized hierarchies of research community based on the relationship can help us get a better insight of the community. We can further study the topic evolution and the collaboration patterns of each individual researcher.

The results of advisor-advisee can also improve the application of Bole search[?] and the expert finding. We use the Bole search problem as the example to demonstrate it.

## 6. CONCLUSION AND FUTURE WORK



In this paper, we study the problem of advisor-advisee relationship mining as an attempt to discover hidden semantic knowledge contained in the links. We propose a two-stage framework to transform the collaboration network step by step, until we construct the advising hierarchy with ranking. In the first stage, we extract the feature of advising relation according to two basic assumptions and construct a candidate graph. In the second stage, we propose Time-constraint Probabilistic Factor Graph (TPFG) model to formalize the problem in a probabilistic ranking model. Further, An efficient learning algorithm is proposed to infer the TPFG model. Experimental results on DBLP data sets demonstrate that the proposed approach can effectively discover the advisor-advisee relationship merely according to collaboration history and primary assumptions. The proposed learning algorithm for TPFG also has a good scalability performance. We show that by making a small set of simple assumptions on the network, one is able to extract certain semantic knowledge hidden behind the links.

Interesting problems related to the approach include how to extend the approach to general relationship mining, to incorporate prior information to enable semi-supervised learning and to cope with multi-typed nodes and links, etc. Other work can be done based on the discovered advisor-advisee relations. For example, we can do hierarchical clustering. Collaboration pattern of individual researchers can be studied, to characterize the growing pattern of a researcher. Along with this path we can move on to study research topic evolution and the development of a research community. Another interesting problem is to correlate the discovered latent relation with social influence analysis. To sum up, there is a lot more for people to exploit the power of the information network and explore inherent knowledge from it.

## 7. ACKNOWLEDGMENTS

The authors would like to thank \*\*\* for his help in \*\*\*.

## 8. ADDITIONAL AUTHORS

## 9. REFERENCES

- [1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.
- [3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [4] H. B. Coonce. Computer science and the mathematics genealogy project. *SIGACT News*, 35(4):117–117, 2004.
- [5] B. Coppola, A. Moschitti, and D. Pighin. Generalized framework for syntax-based relation mining. In *ICDM*, pages 153–162, 2008.
- [6] B. J. Frey. *Graphical models for machine learning and digital communication*. MIT Press, Cambridge, MA, USA, 1998.
- [7] J. Geller and R. Scherl. Challenge: Technology for automated reviewer selection.
- [8] S. P. Kadri, S. Pradhan, K. Hacioglu, W. Ward, J. H. Martin, and D. Jurafsky. Semantic role parsing: Adding semantic structure to unstructured text. In *ICDM*, pages 629–632, 2003.
- [9] 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.
- [10] A. K. McCallum. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002.
- [11] F. Menczer. Combining link and content analysis to estimate semantic similarity. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 452–453, New York, NY, USA, 2004. ACM.
- [12] F. Rinaldi, G. Schneider, K. Kaljurand, M. Hess, and M. Romacker. An environment for relation mining over richly annotated corpora: the case of genia. *BMC Bioinformatics*, 7(Suppl 3):S3, 2006.
- [13] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 797–806, New York, NY, USA, 2009. ACM.
- [14] T. Wu, Y. Chen, and J. Han. Association mining in large databases: A re-examination of its measures. In *PKDD*, pages 621–628, 2007.
- [15] Z. Yang, J. Tang, B. Wang, J. Guo, J. Li, and S. Chen. Expert2bole: From expert finding to bole search. In *Proceeding of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'09)*, 2009.
- [16] X. Yin, J. Han, and P. Yu. Object distinction: Distinguishing objects with identical names. In *Proceedings of IEEE 23rd International Conference on Data Engineering (ICDE'2007)*, pages 1242–1246, 2007.
- [17] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1048–1052, New York, NY, USA, 2007. ACM.
- [18] X. Zhou, X. Pan, and Y. Ren. Web mining of relations from xml and construct database schema. In *CIMCA '06: Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce*, page 211, Washington, DC, USA, 2006. IEEE Computer Society.