Advisor-advisee relationship mining from dynamic collaboration network

Chi Wang
Dept. of Computer Science
University of Illinois at Urbana-Champaign

chiwang1@uiuc.edu

Abstract

In academic world, people are interested in how researchers are connected to each other and how the research community is formed by each individual researcher. As a first step, identifying advisor-advisee relationship can help us answer these questions. Given a collaboration network of researchers, this 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 collaboration data, we need develop a technique to analyze the network evolving with time.

This project aims to model the dynamic collaboration network in order to discover the advisor-advisee relationship between coauthors. 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 for each potential pair of coauthors in this relationship. 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 discovered relations can be further used to analyze the research community evolution. And the methodology can be generalized to mine the roles or relations in a evolving network.

1 Introduction

Nowadays, people and entities are connected in all kinds of information networks. With the rapid growth of online networking applications, such as Facebook, Twitter and MySpace, it is recognized that there is abundant information contained in either online or real-world networks. For example, on Facebook users share their ideas and experiences with friends; on Twitter.com, digg.com, etc. people follow others' news according to their interest and concern. Thus the information network is a good indication how information, ideas and influence are spread away. With the help of link mining technique, people are able to extract knowledge such as community structure and authoritative sources can be discovered from the network data in which links play an important role. However, existing work mainly focus on mining knowledge based on links other than mining knowledge tied in the links themselves. To be more specific, links can be refined, classified and distinguished though in the data they are not. For instance, in a forum where replied messages can be seen as links connecting different users within the same discussion board, these links can have very different meanings: some are supportive while some are opposed. To differentiate them help us better understand the network structure. Once the semantic meaning of the links is extracted from the mingled data, information network analysis will be facilitated in at least three ways. First, the network can be finely modeled because additional information is available other than plain links. Second, hierarchies, clusters and components discovered by different means can be compared to see if they are meaningful. Last, it enriches graph summary and influence analysis.

In this work, we focus on mining the advisor-advisee relationship between coauthors from collaboration network. Collaboration network is a graph composing researchers as nodes, and their collaboration as edges. From the view of knowledge discovery, people are interested in how researchers are connected to each other and how the research community is formed by each individual researcher. As a first step, identifying advisor-advisee relationship can help us answer these questions. The community evolution is motivated by the development of each individual. If we can figure out how each researcher grows from an advisee to a self-governed researcher or even an advisor, not only we can position each person in a chronological axis in a correct order, but also we can sketch the whole community in a very clear view. We can further do clustering, influence analysis and research topic evolution, etc.

Many projects have been set up to maintain such information for various research fields. These include the Mathematics Genealogy Project[4], the Computer Engineering

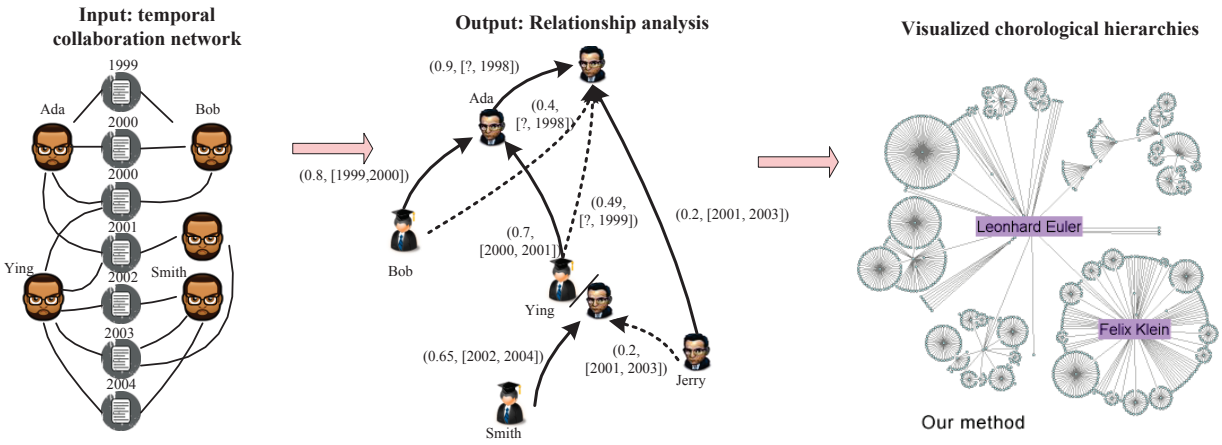


Figure 1: Example of advising relationship analysis on the co-author network.

Academic Genealogy, the AI Genealogy Project and the Software Engineering Academic Genealogy. However, all of these projects rely on manually collecting the academic genealogy data which makes them quite costly. For a given collaboration network, it is not always the case we can easily find corresponding dissertation data. Either lack of data or difficulty to identify the mapping will lead to failure. Therefore, we need to develop a general analyzing technique in order to automatically mine the relations from the network data.

Using graph mining approach, nodes or edges with certain properties such as having the largest centrality or betweenness can be discovered. We can also compute importance of a node, and relevance of neighboring nodes, e.g. using PageRank[2]. Furthermore, we can do ranking and clustering based on the link information, e.g. using NetClus[**?**]. However, with all the existing method, it is still difficult to differentiate the social role from the static collaboration network. We must consider the temporal information and build a unified model for the dynamic collaboration network. The methodology is expected to apply for general dynamic network.

There is a vast amount of collaboration information network available online. We use the DBLP Computer Science Bibliography Database maintained by Michael Ley. The DBLP data consists of 1 million distinct publications of over 650, 000 authors. Each publication is associated with an ordered list of authors, the year of publication and other information such as the venue where it is published. In Figure 1, the left figure shows an example of coauthor network. The collaboration along different time are mixed in one heterogenous network and the publication year is labeled on each paper. Our analysis takes the heterogenous 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 generated dynamic social influence scores, we can consider to improve applications such as research community detection, evolution analysis and automated reviewer selection[6]. The right figure gives an example of visualized chorological hierarchies.

The problem of relationship mining is quite different from existing works on information network analysis and poses a set of unique challenges.

- *Latent relation.* The advisor-advisee relation is completely hidden in the collaboration data. There is no explicit sign 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. Specifically, the advisor of each author and the advising time of all of them are modeled together as a joint probability of as many hidden variables as authors with time constraint. An efficient algorithm to optimize the joint probability and obtain ranking score is designed as a process of message propagation on the network.

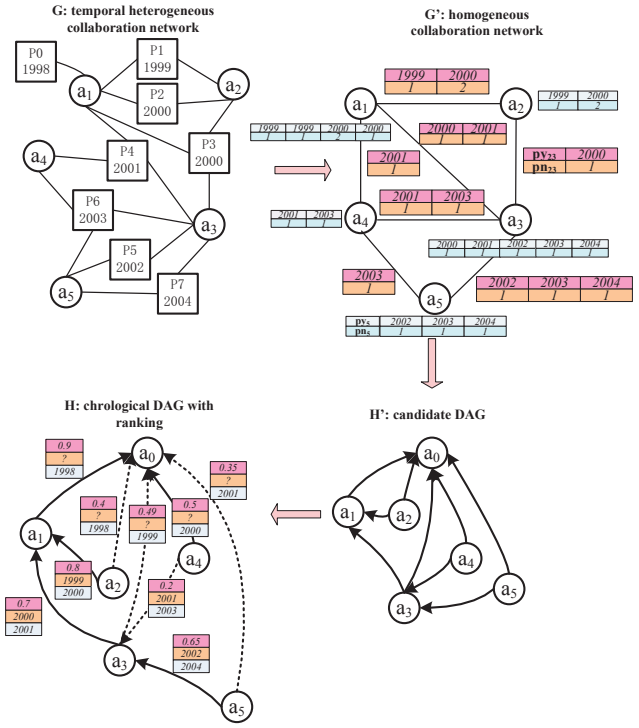


Figure 2: Example of graph transformation.

The rest of the paper is organized as follows: Section 2 formally formulates the problem. Section 3 explains the proposed approach. Section 4 presents experimental results that validate the computational efficiency and efficacy of our methodology. Finally, Section 5 concludes and discusses future work.

2 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 heterogeneous collaboration networks $\{G\} = \{(V = V^p \cup V^a, E)\}$, where $V^p = \{p_1, \ldots, p_{n_p}\}$ is the set of publications, with p_i published in time t_i, $V^a = \{a_1, \ldots, a_{n_a}\}$ 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 $V' = \{a_0, \ldots, a_{n_a}\}$ 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}. Although our original motivation is to find PhD advisors, in reality, this problem is more complicated than finding one doctorial advisor from all the coauthors of a given researcher because: first, one could have multiple advisors like master advisors, PhD co-advisors, post-doctorial 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 advisors 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 $\imath$ is denoted Y_i, 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 chorological 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 θ is a threshold such as 0.5. We use $P@(k, \theta)$ to denote this method. It is predictable that large k and large θ leads to better recall and worse precision. How to choose k and θ 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.

3 Approach

We have formalized the problem and defined the transformation process to solve the problem. In this section we describe the framework of advisor-advisee relationship mining and our approach in each process. 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.

3.1 Assumptions and Framework

We have the following assumptions based on common sense.

Assumption 1

$$\forall 1 \leq x \leq n_a, ed_{y_x} < st_x < ed_x \tag{1}$$

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, $t_1 < t_2$. In one word, one must graduate before he could advise.

Assumption 2

$$\forall 1 \leq x \leq n_a, \mathbf{py}^1_{y_x} < \mathbf{py}^1_x \tag{2}$$

That means for a given pair of advisor and advisee, the advisor always has a longer publication history than the advisee. $\mathbf{py}^1_x$ represents the first component of vector $\mathbf{py}_x$.

Assumption 1 is an important time constraint which determines the correlation between one's advisor and the advisor of his neighbor. For example, in Figure 2, a_4 coauthors two papers with a_3, which is more than with other researchers. However, since a_3 is highly probable to be advised by a_1 in the year 2001, when a_4 starts to collaborate with him, he is not likely to advise a_4 since that time. As a result, 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 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 and asymmetry. For transitivity, if $a_1 \rightarrow a_3$ is a possible pair of advisor-advisee, and so is $a_3 \rightarrow a_5$, then $a_1 \rightarrow a_5$ will be a possible pair while $a_5 \rightarrow a_1$ will not. 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 3.2. 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'. After the first stage, we have already been able to predict the advising relation by selecting the local optimal. However, to model the network in a integrated way and achieves the global optimal we need to take the second stage.

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.

We will describe our approach in these two stages in following sections.

3.2 Stage 1: Preprocessing

The purpose to do preprocessing is to generate the candidate graph H'. On one hand, we hope the candidates of each author's advisor are as few as possible to reduce the search space. On the other hand, we need to make sure the real advisor is not excluded from the candidate pool in most cases. First we need to integrate the heterogeneous collaboration information into a homogeneous author network G'. Then we can keep only those edges indicating possible advising relations from G', and assign them a direction. To generate G', we need to process 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 check whether $\mathbf{py}^1_j + 2 > \mathbf{py}^1_{ij}$. If so, according to a corollary of Assumption 1, we know that when a_i and a_j started to cooperate, a_j had not graduated, thus excluded. Next, we compute Kulczynski measure of the publications of them for each year

from the time they started cooperation to the end of their cooperation. It is defined as

$$kulc_{ij}^t = \frac{\sum_{\mathbf{py}_{ij}^k \le t} \mathbf{pn}_{ij}^k}{2} \left(\frac{1}{\sum_{\mathbf{py}_i^k \le t} \mathbf{pn}_i^k} + \frac{1}{\sum_{\mathbf{py}_j^k \le t} \mathbf{pn}_j^k}\right) \tag{3}$$

The Kulczynski measure reflects the correlation of the two authors' publication. [9] shows that there usually exists high correlation between the total publications of advisors and advisee. Here we further incorporates the time factor, to calculate the measure year by year, and see whether there exists some year when the measure is above a lower bound θ. We take this lower bound small enough to only filter out impossible advisor-advisee pairs. 0.05 is a safe value for θ in practice. Besides the lower bound condition, we also check whether there is a increase in the sequence $\{kulc_{ij}^t\}_t$. If not, the correlation between the two authors keeps decreasing since they start collaboration and that does not support their advising relation.

We calculate another measure IR in the corresponding year as the Kulczynski measure sequences. IR [9] is used to measure the imbalance of the occurrence of a_j given a_i and the occurrence of a_i given a_j. It is formally calculated as follows.

$$IR_{ij}^t = \frac{\sum_{\mathbf{py}_j^k \le t} \mathbf{pn}_j^k - \sum_{\mathbf{py}_i^k \le t} \mathbf{pn}_i^k}{\sum_{\mathbf{py}_i^k \le t} \mathbf{pn}_i^k + \sum_{\mathbf{py}_j^k \le t} \mathbf{pn}_j^k - \sum_{\mathbf{py}_{ij}^k \le t} \mathbf{pn}_{ij}^k} \tag{4}$$

If the first value in the IR sequence is negative, that indicates a_i publishes more papers than a_j before their collaboration. In this case we take a_j out of consideration to be a_i's advisor.

When the pair passes the test of all these conditions, 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}. The starting time st_{ij} is estimated as the time they started to collaborate, while the ending time ed_{ij} is estimated as the time point when the Kulczynski measure starts to decrease consecutively. This is intuitive since when an advisor graduates, the correlation with his advisor is likely to decrease during the interim. During the advising time the measure may drop temporarily, so we demand the decrease to last for a period in order to estimate the graduation time as accurate as possible. In practice, the period length is set to be 3 years, except when the decrease has happened over 6 times before.

After we estimate st_{ij} and ed_{ij}, we calculate the average of Kulczynski and IR measure during that period, and average the two as the local likelihood. The intuition is that the more correlated their publication is, and the more the tendency of collaboration is from a_i to a_j, the more likely a_j is a_i's advisor. So we have the following formula.

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

That finishes the processing for one edge e_{ij} of G'. 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'.

3.3 Stage 2: TPFG Model

From the candidate graph H' we know the potential advisors of each author and the likelihood based on local information. However, merely based on local information it is hard to tell who is one's real advisor from all the candidates. In the network people have complicated interrelationship and dynamic social roles. 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 TPFG 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.

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

with

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

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

Equation 7 is the constraint according to Assumption 1. If any configuration of $\{y_i, st_i, ed_i\}_{a_i \in V^a}$ violates it, the joint probability of that configuration is 0. To find most probable values of all the hidden variables, we need to maximize the joint probability of all of them. Therefore, it becomes a optimization problem of Eq. 6 under the constraint (7).

It is straight 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 is $(CT^2)^{n_a}$. That is exponential. It is intractable to do exhaustive search. On the other hand, if we take local maximal of each feature function $g(y_i, st_i, ed_i)$, i.e. let

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

then it typically violates Eq. 7 and achieves global minimum. However, as long as Eq. 7 is satisfied, increase of local feature function value will also increase the joint probability. By this intuition 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.,

$$\{st_i, ed_i\} = arg \max_{st_i < ed_i} g(y_i, st_i, ed_i) \quad (9)$$

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 precompute the best advising time as st_{ij} and ed_{ij} for each $y_i = j$. This simplification is reasonable since our major goal is to find the advisor. Also it is unlikely we can accurately decide the advising time, but a rough estimate is enough if we successfully identify the relation. Now only $\{y_i\}$ are variables to optimize, though their values are correlated due to Eq. 7. If we combine the constraint Eq. 7 into the feature function, the objective function becomes

$$P(y_1, \ldots, y_{n_a}) = \frac{1}{Z} \prod_{i=1}^{n_a} f_i(y_i | \{y_x | x \in Y_i^{-1}\}) \quad (10)$$

with

$$f_i(y_i = j | \{y_x | x \in Y_i^{-1}\}) = g(y_i, st_{ij}, ed_{ij}) \prod_{x \in Y_i^{-1}} I(y_x \neq i \vee ed_{ij} < st_{xi}) \quad (11)$$

where

$$I(y_x \neq i \vee ed_{ij} < st_{xi}) = \begin{cases} 1 & y_x \neq i \vee ed_{ij} < st_{xi} \\ 0 & y_x = i \wedge ed_{ij} >= st_{xi} \end{cases} \quad (12)$$

is the identity function. It is clear that if any author a_x is advised by a_i and their advising time conflict, the function takes 0; otherwise it takes 1. 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. 10. Furthermore, if we want to compute the rank score of each advising relationship, e.g. $a_j \xrightarrow{\text{advise}} a_i$, we just need to compute the conditional maximal probability

$$r_{ij} = \max P(y_1, \ldots, y_{n_a} | y_i = j) \quad (13)$$

This simplification assures that for each configuration of $\{y_i\}$, the simplified optimization achieves either 0 or the conditional optimal given that configuration. The search space size now becomes C^{n_a}, reduced but still exponential. Exhaustive search is still impracticable. However, 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 TPFG 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 are all of 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

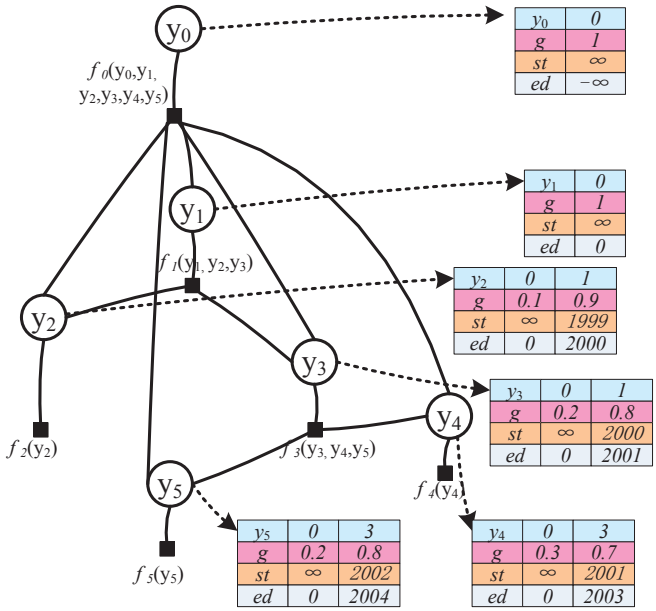


Figure 3: Graphical representation of the time-constrained probabilistic factor graph. $\{y_0, \ldots, y_5\}$ are hidden variables defined on all nodes; $f_i(.)$ represents a factor function defined on a hidden variable and its potential advisee sets as neighbors.

one edge between one variable node y_x and a function node $f_i(.)$ if and only if $f_i(.)$ 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 $f_1(y_1, y_2, y_3)$. We can see that two hidden variables are correlated iff. their corresponding author nodes are linked by an edge on the candidate graph H', which means there is a potential advising relationship between them. And once a variable y_i changes its value, it will affect the value of all the functions corresponding to the potential advisor and advisee sets $Y_i \cup Y_i^{-1}$.

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. There are various ways to estimate their values. Here we use the method as described in Section 3.2, and 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 TPFG 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.

3.4 Model Learning

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

need to infer the marginal maximal joint probability on TPFG , according to Eq. 13. We first introduce the algorithm for general factor graph, discuss its deficiency and then propose our algorithm.

Baseline: sum-product + junction tree There is a general algorithm called *sum-product* [7] 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. 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 function. To compute the marginal maximal probability, we just need to change sum-product to max-sum with a logarithmic transformation of the function value. If we treat TPFG as a tree-structured factor graph, the message passing rule will be:

$$
\begin{aligned}
m_{y_i \to f_j()}(y_i) &= \sum_{j' \in Y_i \cup \{i\}, j' \neq j} m_{f_{j'}() \to y_i}(y_i) \\
m_{f_j() \to y_i}(y_i) &= \max_{\sim \{y_i\}} (\log f_j(y_i, \{y_{i'}\}) + \\
&+ \sum_{i' \in Y_j^{-1} \cup \{j\}, i' \neq i} m_{y_{i'} \to f_j()}(y_{i'}))
\end{aligned}
$$

where $j' \in Y_i \cup \{i\}, j' \neq j$ represents $f_{j'}()$ is a neighbor node of variable y_i on the factor graph except factor $f_j()$, $\sim \{y_i\}$ represents all variables in $Y = \{y_1, \ldots, y_{n_a}\}$ except y_i. In exact inference, when the passing finishes, for any $j \in Y_i$, the sum of $m_{y_i \to f_j()}(y_i)$ and $m_{f_j() \to y_i}(y_i)$ actually corresponds to a logarithm of conditional maximal probability for $y_i = j$.

Unfortunately TPFG contains cycles. This algorithm cannot be applied directly. Some nodes will always stay idle if we use the same message passing scheme. One solution to generalize it is an exact inference procedure known as *junction tree algorithm*[1]. It first triangulates the original dependency graph and then constructs a new tree-structured undirected graph called a *join tree*, whose nodes correspond to the maximal cliques of the of the triangulated graph, and whose links connect pairs of cliques that have variables in common. If the tree is condensed so that any clique that is a subset of another clique is absorbed into the large clique, this gives a *junction tree*, which can be solved by sum-product. The junction tree is exact for arbitrary graphs. Unfortunately, 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 selection of which pairs of cliques to connect is important and is done as to find a *maximal spanning tree*. The process to construct a junction tree itself consumes long time and large memory for our TPFG of the large scale with more than 600,000 variables.

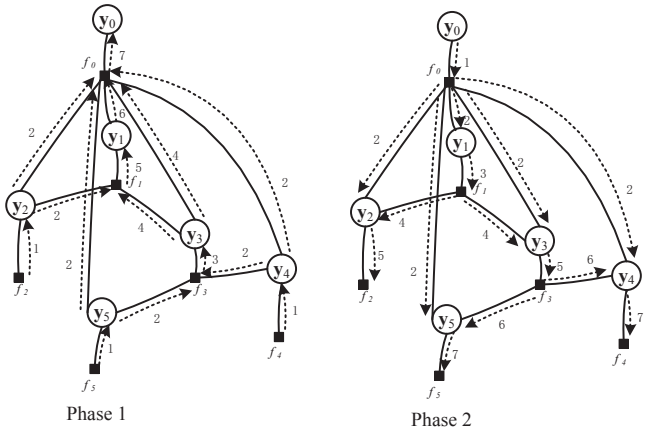


Figure 4: The 2-phase message passing schema.

To reduce the computational cost, we can do approximate inference instead of exact inference. One approach *loopy belief propagation*[5] simply applies the sum-product algorithm in cycle-containing graph. This is possible because the message passing rules are purely local. However, because the graph now has cycles, information can flow many times around the graph, and we need to define the message passing schedule. That idea inspires us to design the following algorithm according to the special property of TPFG.

New TPFG 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 TPFG some nodes with be waiting forever due to the existence of cycles. For the example in Figure 3, at the beginning $f_2(), f_4(), f_5(), y_0$ could send message to $y_2, y_4, y_5, f_0()$ respectively because they are of degree 1. Then the message passing process will be stuck because each node is waiting for at least two messages. To overcome this problem, we arrange the message passing in the a mode according to 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: $m_{f_i() \to y_i}, m_{y_i \to f_j()}$ 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: $m_{y_i \to f_i()}, m_{f_j() \to y_i}, j \in Y_i$, 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 as follows.

$$m_{f_i()\to y_i}(x) = \max_{st_{ki}>ed_{ix},\forall y_k=i}(\log l_{ix} + + \sum_{k'\in Y_i^{-1}} m_{y_{k'}\to f_i()}(y_{k'})) \quad (14)$$

$$m_{y_i\to f_j()}(x) = m_{f_i()\to y_i}(x) \quad (15)$$

$$m_{y_i\to f_i()}(x) = \sum_{j\in Y_i} m_{f_j()\to y_i}(x) \quad (16)$$

$$m_{f_j()\to y_i}(x) = \max_{st_{kj}>ed_{jy_j},\forall y_k=j}(\log l_{jy_j} + m_{y_j\to f_j()}(y_j) + \sum_{k'\in Y_j^{-1},k'\neq i} m_{y_{k'}\to f_j()}(y_{k'})) \quad (17)$$

After the two phases of message propagation, we can collect the two messages on any edge and obtain the marginal function. For example,

$$r_{ij} = \max P(y_1,\ldots,y_{n_a}|y_i=j) = \exp\left(m_{f_i()\to y_i}(j) + m_{y_i\to f_i()}(j)\right) \quad (18)$$

However, this algorithm still has redundant computation. The messages sent between function nodes and variables nodes are function values, which need to be stored as vectors. Some messages are never used during the final merge, and some messages are simply transmitted from one variable node to its corresponding function node. Based on all of these observations, we further simplify the message propagation. We eliminate the function nodes and the internal messages between a function node and a variable node, but transform 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 two vectors, one sent and one received. In this way both time and space are saved.

The improved message propagation is still separated into two phases. In the first phase, the messages $\mathbf{sent}_i$ which are messages passed from one to their ascendants are generated in a similar order as before. In the second, messages returned from ascendants $\mathbf{recv}_i$ are stored in each node. After the two phases, each node collects the two vectors to generate the final ranking score. The rules are formally defined as follows. (Derivation of the update rule is omitted.)

$$\mathbf{sent}_{ij} = \log l_{ij} + \sum_{k\in Y_i^{-1}} \max_{st_{kx}>ed_{ij} \text{ or } x\neq i} \mathbf{sent}_{kx} \quad (19)$$

$$\mathbf{recv}_{ij} = \max_{j'\in Y_j, ed_{jj'}<st_{ij}} (\mathbf{recv}_{jj'} + \log l_{jj'} + + \sum_{k\in Y_j^{-1},k\neq i} \max_{x\in Y_k, st_{kx}>ed_{jj'} \text{ or } x\neq j} \mathbf{sent}_{kx}) + \sum_{x\in Y_i,x\neq j} \max_{j'\in Y_x} (\mathbf{recv}_{xj'} + + \sum_{k\in Y_x^{-1},k\neq i} \max_{x'\in Y_k, st_{kx'}>ed_{xj'} \text{ or } x'\neq x} \mathbf{sent}_{kx'}) \quad (20)$$

$$r_{ij} = \exp(\mathbf{sent}_{ij} + \mathbf{recv}_{ij}) \quad (21)$$

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 $\mathbf{sent}_{ij}$ 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. $count_x = \emptyset$;
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 $\mathbf{sent}_{ij}$ according to Eq. 19;
    $count_j - -$;
    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**
    $\mathbf{recv}_{j0} \leftarrow 0$
  end
  else
    foreach $j' \in Y_j$ **do**
      Collect $\mathbf{recv}_{jj'}$ and $\mathbf{sent}_{jj'}$ to compute $r_{jj'}$ according to Eq. 21 and prepare to compute $\mathbf{recv}_{ij}$;
    end
  end
  foreach $i \in Y_(^{-1} j)$ **do**
    Compute $\mathbf{recv}_{ij}$ according to Eq. 20;
  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 TPFG 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 $\mathbf{sent}_i$ 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 $\mathbf{recv}_i$. 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 $O(n_a)$.

4 Experimental Results

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

4.1 Experiment Setup

Data Sets and method We perform experiments on several real-world data sets. We use the DBLP Computer Science Bibliography Database 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[1]. We refer to them as MAN and MathGP respectively. On the web site of a certain person, we can find information of PhD students, MS students and post-doc scholars and collaborators of that advisor. Mathematics Genealogy contains abundant graduated PhD and their advisor. However, a large fraction is not from Computer Science domain. Of the remaining many are too old beyond the publication records in our database. Besides, only a part of the names in their project match with DBLP. Therefore, although we crawled 50,000 pairs from Mathematics Genealogy, only around 2000 of them can be used for our test. 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 hundreds of colleague pairs which are negative samples for advisor-advisee relationship. Teacher, PhD and MathGP are used as ground truth, while Colleague is used as ground falseness.

One thing to note is that even in the ground truth, one may have other advisor than the labeled one. For example, Nicholas Roy has two co-advisors Sebastian Thrun and Tom Michael Mitchell, but we only label the first one. We actually do not find this fact until we run the experiment and the result shows that both are ranked top 2 as Roy's advisor. There are many similar cases. For this reason, we do not simply predict the advisor-advisee relation as taking the top 1 ranked candidate. We extend it a little more to include top k candidates, with $k \leq 3$, since it is rare a person would have more than 3 advisors. To predict an cooperator a_j as a_i's advisor, we require a_j is ranked top k among all a_i's advisor candidates, including a_0 which indicates a_i has no advisor. It is simple but effective in practice. We can expect that the recall of one's advisor will increase while the precision will decrease, if we increment k.

We compare our improved TPFG inference algorithm (will be referred as TPFG 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 TPFG does. The difference is that JuncTcomputes the exact joint probability as the ranking score while LOCALadopts 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 can use Teacher plus a subset of Colleague as training data set.

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 networking 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[8], which is an integrated collection of Java code for machine learning. We implement TPFG with Microsoft Visual C++ 2008. And we use LIBSVM[3] to perform SVM training and prediction.

4.2 Accuracy

We conduct a series of experiments to explore the capability of TPFG algorithm in mining advisor-advisee relationship in the network. First, as we mentioned in Section 3.1 and Section 3.2, different assumptions about advising relationship can be tested to see which of them are the best combination to reflect the reality. For example, in the filtering process, the rules to estimate the graduation year are flexible and the effect of them can be tested through the prediction results. Second, we extract small fraction of the whole DBLP network and feed as the original input G to TPFG , 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 TPFG can be further improved by utilizing training data.

4.2.1 rules validation

By selecting different rules based on various assumptions, the preprocessing stage of our approach will generate different candidate graph H. These rules reflect intuitive knowledge about the advisor-advisee relations other than the Assumption 1 and Assumption 2. It is a practical question that which of the rules best fit the data in the sense of sufficient generality and specialty. By experiment we can find out the best rules for a given data set. We try 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

[1] http://www.genealogy.math.ndsu.nodak.edu/

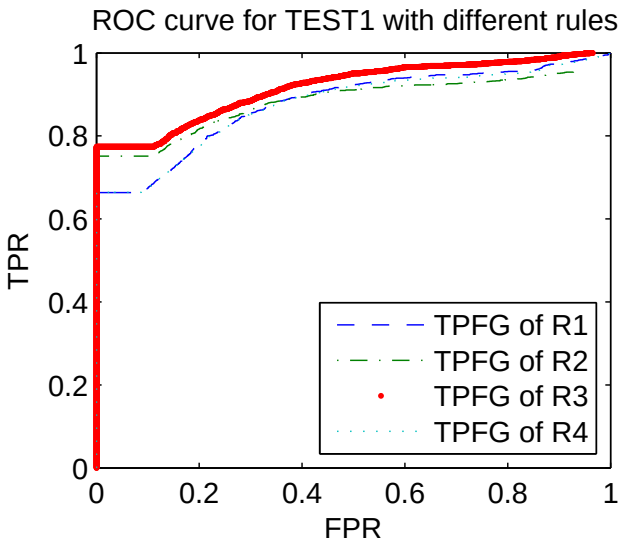
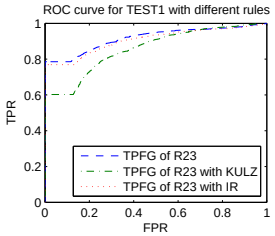
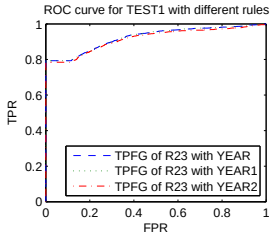


Figure 5: ROC curves for TPFG with different rules



(a) ROC curves for TPFG with different evaluation measures for l



(b) ROC curves for TPFG with different estimation method for ed

Figure 6: rule refinement.

output of the algorithm TPFG . Then we sort these ranking score in a descendant order, and draw the ROC curve for them.

From Figure 5 we can see that R2 and 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. From now on we use R2 and R3 as filtering rules, use the combination of Kulczynski and IR as local likelihood evaluation measure and use YEAR2 as the graduation year estimation method if not mentioned specifically.

4.2.2 boosts by the network

TPFG algorithm relies on the network structure to propagate the message and estimate the global optimal joint probability. LOCAL consider for every individual node the local feature only. Now we examine the power of the network by feeding TPFG with different scales of subnetwork and compare the accuracy. Given a set of pairs of authors we want to test, we can construct subgraphs containing the nodes interested and remove the portions of the whole network

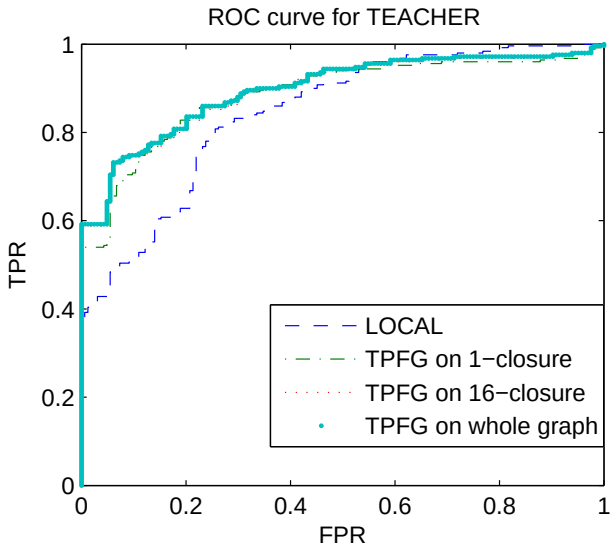


Figure 7: ROC curves for TPFG on different closures

which are seemingly far away from them. 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. When we do the DFS, we exclude the node a_0 because it is a virtual node and connects to every node. DFS-1 generates 1-closure, including the given set of nodes and the direct neighbors of them, which are no more than direct coauthors of them. When d increases, the subnetwork grows larger. The growth will ends when the subnetwork is already the the complete closure, i.e. the maximal connected subgraph of H containing the given set. We run TPFG for these closures and draw the ROC curve for them.

From Figure 7 we see that with different scales of closures provided, TPFG achieves better accuracy when the depth increases, and they all outperform the LOCAL method. And on the complete closure TPFG achieves the same accuracy as on the whole network since the disconnected components will not affect each other. The complete closure is actually the largest component whose size is shown in Table **??**.

On these various scaled subnetworks, TPFG achieves different level of approximation to the optimal global joint probability on the whole network. Although we are not able to know the exact optimum on the whole network, we are able to know the exact maximal joint probability on a small subnetwork. We construct 1-closure for a small data set, and then run TPFG and JuncTon the subnetwork with 1310 nodes to compare the prediction performance. From Figure **??** we find that in the small graph TPFG approximates well with the exact inference algorithm JuncT. Unfortunately JuncTfails on larger network due to the high requirement for the space. To further validate the performance we need to compare with approaches based on other models.

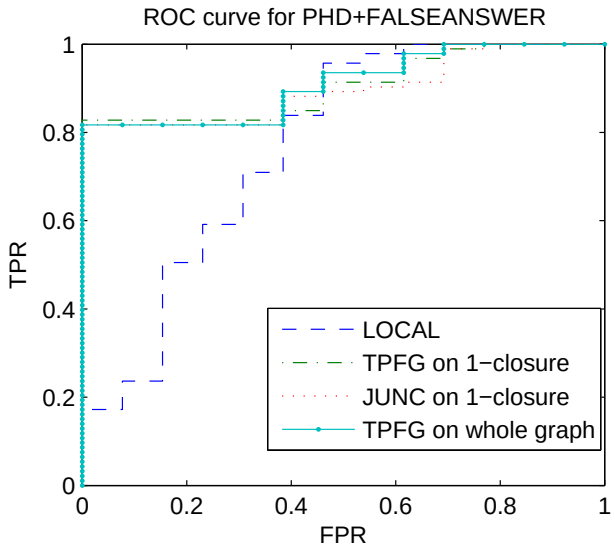
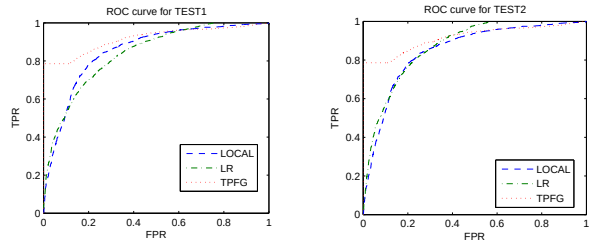


Figure 8: ROC curves for Junc and TPFG on smaller graphs



(a) ROC curves for LOCAL, TPFG and LR on TEST1

(b) ROC curves for LOCAL, TPFG and LR on TEST2

Figure 9: Comparison with training based method.

4.2.3 approach based on training

SVM is a special form of Linear Threshold 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 problem to a classification on the ordered pairs (a_i, a_j). In this setting it requires to define some features for each pair of coauthors, and train the classifier by feeding both positive and negative samples. To make sure the classifier make use of the same information as in our probabilistic model, we define following features based on the number of papers and publication year.

Since 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 need to know the ranking score of each pair to further compare the results. Thus we train a Linear Threshold (LR) model according to the same features and training data, calculate the probabilistic score for each pair in the test data, and rank them from high to low to draw the ROC curve. We can also compare the prediction accuracy by P@k since we now can calculate the score for all of one's advisor candidates.

Although in this paper we restrict our model as a unsupervised learning approach, it can also be modified to achieve supervised learning. Here we do not discuss that part but gives a simpler method to incorporate labeled data when we do the prediction with the output of ranking score of our algorithm. If labeled data are given as truth, we can adjust the parameter θ 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.

data set	SVM	LOCAL	TPFG
TEST1_POS	84.2%	71.5%	72.3%
TEST1_NEG	52.1%	85.7%	86.6%
TEST1_ALL	63.0%	80.4%	81.3%
TEST2_POS	39.7%	71.2%	71.9%
TEST2_NEG	83.2%	85.5%	86.3%
TEST2_ALL	68.4%	80.1%	80.9%

Table 1: Accuracy of prediction.
TRAIN1=Colleague(491)+PHD(100).
TEST1=Teacher(257)+MathGP(1909)+Colleague(2166).
TRAIN2=Teacher(257)+Colleague(2166).
TEST2=PHD(100)+MathGP(1909)+Colleague(4351).

data set	LR	LOCAL before	LOCAL after	TPFG before	TPFG after
TEST1	73.4%	77.3%	77.7%	71.4%	82.2%
TEST2	74.6%	77.3%	77.7%	71.2%	81.8%

Table 2: Accuracy of prediction.
TRAIN1=Colleague(491)+PHD(100).
TEST1=Teacher(257)+MathGP(1909)+Colleague(2166).
TRAIN2=Teacher(257)+Colleague(2166).
TEST2=PHD(100)+MathGP(1909)+Colleague(4351).

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

4.2.4 case study

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

Table 3 shows some inconsistent results by different methods.

author	MathGP	LOCAL	TPFG
Maren Bennewitz	Wolfram Burgard	Sebastian Thrun	Wolfram Burgard
Tatiana Surazhsky	Gershon Elber	Gershon Elber	Gill Barequet
Jiangzhuo Chen	Chris Peikert	Rajmohan Rajaraman	Chris Peikert

Table 3: Some inconsistent case study

4.3 Scalability Performance

We evaluate the efficiency of TPFG , on the two candidate graphs. Since the statistics of them do not differ much and neither does the running time, we only shows the result on the larger one. Table 4 lists the running time required for those algorithms. The same filtering procedure is used for JuncT, LOCAL and TPFG . SVM does not need to preprocess the whole DBLP network, thus takes shorter time. The preprocessing time consumes longer time than model inference, except for JuncT. It cannot finish for the whole DBLP network in 24 hours. LOCAL does not do further learning after preprocessing. The training time will become slow if we switch the training data and testing data for SVM. The prediction is the fastest part. For LOCAL and TPFG , since they generate results for the whole DBLP network, reading the results in memory takes a minute.

Considering the scale of our data set, LOCAL and TPFG are proved to be scalable. With regard to the total time, TPFG costs hardly any more than LOCAL. 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 4: Scalability performance of different methods on two stages.

Stage	JuncT	LOCAL	SVM	TPFG
preprocessing	30min	30min	20min	30min
learning	>24hr	N/A	1min	1min
prediction	1min	1min	10s	1min

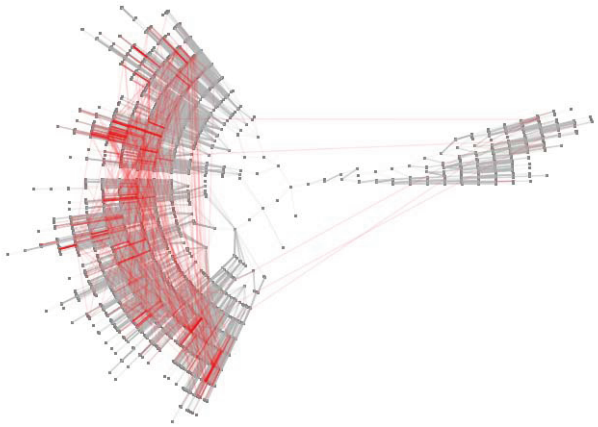


Figure 10: visualized advising relation

4.4 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.

5 Conclusion and Future Work

In this paper, we study the problem of advisor-advisee relationship mining. We propose a two-stage framework to transform the heterogeneous collaboration network step by step, until we construct the advising hierarchy with ranking. In the first stage, se extract the feature of advising relation according to two basic assumptions and construct a candidate graph. In the second stage, we proposes 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 apply the advising relation learned by our approach to help Bole search and research commu-

nity evolution.

Interesting problems related to approach include how to extend the approach to general relationship mining, how to incorporate prior information to enable semi-supervised learning. Other future work can be done based on the advisor-advisee relationship results. For example, we can do hierarchical clustering as well as dynamic analysis on it. 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 explore and exploit the power of the information network.

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

[6] J. Geller and R. Scherl. Challenge: Technology for automated reviewer selection.

[7] 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.

[8] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[9] T. Wu, Y. Chen, and J. Han. Association mining in large databases: A re-examination of its measures. In *PKDD*, pages 621–628, 2007.