# Mining Advisor-Advisee Relationships from Research Publication Networks

Chi Wang[‡], Jiawei Han[‡], Yuntao Jia[‡], Jie Tang[†], Duo Zhang[‡], Yintao Yu[‡], Jingyi Guo[†]

[‡]University of Illinois at Urbana-Champaign, USA [†]Tsinghua University, China

Email: {chiwang1,hanj,yjia3,dzhang22,yintao}@illinois.edu, {jietang,guojy07}@tsinghua.edu.cn

## ABSTRACT

Information network contains abundant knowledge about relationships among people or entities. Unfortunately, such kind of knowledge is often hidden in a network where different kinds of relationships are not explicitly categorized. For example, in a research publication network, the advisor-advisee relationships among researchers are hidden in the coauthor network. Discovery of those relationships can benefit many interesting applications such as expert finding and research community analysis. In this paper, we take a computer science bibliographic network as an example, to analyze the roles of authors and to discover the likely advisor-advisee relationships. In particular, we propose a time-constrained probabilistic factor graph model (TPFG), which takes a research publication network as input and models the advisor-advisee relationship mining problem using a jointly likelihood objective function. We further design an efficient learning algorithm to optimize the objective function. Based on that our model suggests and ranks probable advisors for every author. Experimental results show that the proposed approach infer advisor-advisee relationships efficiently and achieves a state-of-the-art accuracy (80-90%) without any supervised information. We also apply the discovered advisor-advisee relationships to bole search, a specific expert finding task and empirical study shows that the search performance can be effectively improved (+4.09% by NDCG@5).

## Categories and Subject Descriptors

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

## General Terms

Algorithms, Experimentation

## Keywords

Relationship mining, Time-constrained probabilistic factor graph, Coauthor network, Advisor-advisee prediction

## 1. INTRODUCTION

With the rapid growth of the social web, particularly online networking applications such as Facebook, Youtube and Twitter, people are closely connected via different types of relationships. It is well recognized that different types of social relationships have essentially different influence between people, which forms the complex and subtle force that governs the dynamics of social networks. For example, in the social network, a graduate's research topic may be mainly influenced by his advisor; while his living habits may be influenced by his family. Awareness of the relationship types can offer abundantly additional information for many mining applications such as community discovery and expert finding. For example, if we know advisor-advisee relationships between researchers, we can easily discover how researchers form different communities, how research topics have been emerging and evolving in the past years, and how a researcher influences the academic research community.

However, in reality, such information (relationship type) is often hidden in the networks due to different reasons. For example, advisor-advisee relationships are hidden in the coauthor network (*e.g.*, on DBLP); family relationships are hidden in the friendship network (*e.g.*, on Twitter or MSN). Several projects aim to maintain the types of relationships, such as LinkedIn and AI Genealogy. The former requires users to label their professional relationships (e.g., colleagues or advisor-advisee) with each friend and the latter asks human annotators to manually label the advisor information for various research fields. However, these methods heavily rely on manual efforts, which significantly limits its wide use. An ideal solution is to design a method that automatically uncovers the hidden relationship types from the networkNevertheless, it is nontrivial to accurately differentiate social relationships, especially in a real large network. For example, neither the most frequent coauthor nor most authoritative researcher among one's collaborators is assured to be his advisor. For real data it could be difficult even for human beings to tell who is one's advisor by the publication list. Sometimes, in a specific application (e.g., the coauthor network), heuristical rules can be defined to identify the relationship type according to human intuitive assumptions. However, our preliminary experimental result (cf. Section 5) shows that the typical heuristic rules can only achieve an accuracy of 70-80%. Even for multiple rules combined with a supervised learning model trained using different features, the accuracy is still only 80% on average. Moveover, it is often difficult to collect supervised information for training in practice.

In this study, we try to conduct a systematic investigation of the case of mining advisor-advisee relationships between authors in a research publication network. Identification of such advisor-advisee relationships can offer us a chance to better understand the insight of the research community, as it provides additional semantic information on the links other than the simple, explicit coauthor relationships. For example, we can position each person in


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

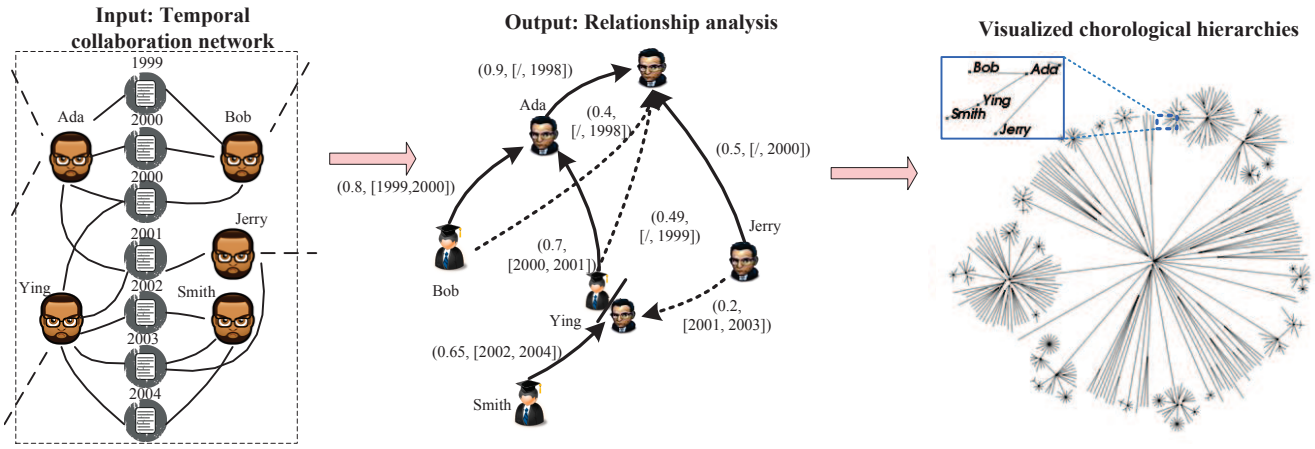



**Figure 1: Example of advising relationship analysis on the co-author network.**

a chronological axis in the right order and sketch the whole community in a clear view. Certain applications of expert finding can also be benefited from the identified advisor-advisee relationships, as people looking for experts may not only care about the personal academic achievements but also are interested in how many "experts" they can foster.

To clearly illustrate the problem, Figure 1 gives an example of advisor-advisee relationship analysis on a research publication network. The left figure shows the input: an temporal collaboration network, which consists of authors, papers, and paper-author relationships. The middle figure shows the output of our analysis: an author network with solid arrow indicating the advising relationship, and dotted arrow suggesting potential but less probable relationship. For example, the arrow from Bob to Ada indicates that Ada is identified as the advisor of Bob. The triple on the edge, i.e., $(0.8, [1999, 2000])$, represents Ada has the probability of 80% to be the advisor of Bob from 1999 to 2000. Such results can benefit many potential applications such as research community detection and evolution analysis. The right figure gives an example of visualized chronological hierarchies. The parent-child relation in the tree corresponds to the advisor-advisee relationship. We can see the advising path from root to leaf.

The problem we study is rather different from existing relevant research (*e.g.*, relation mining). Our work analyzes links rather than text or labeled annotations which poses a set of challenges.

- *Latent relation.* The advisor-advisee relationship is hidden in the network. There is no supervised information 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 insufficient to apply simple rules without considering the inherent correlation of network. When the search space becomes exponential in size, it is important to develop a method that can scale well to real large networks.

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 period are modeled together as a joint probability of as many hidden variables as authors. We further design an efficient algorithm to optimize the joint probability via a process of message propagation on the network. By experiments we show this unsupervised approach can achieve an accuracy of 80-90%, leading by 5-20% against several baseline methods. We also apply the identified advisor-advisee relationships to bole search (best supervisor finding) and demonstrate that the performance of bole search can be clearly improved (+4.1%). The proposed framework is generalizable to other applications. For discovering other type of relationships, the additional requirement is to redefine the feature function and the potential constraints.

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 the study and discusses the future work.

## 2. RELATED WORK

This work is different from the existing study in *Relation Mining* and *Relational Learning*. Previous studies in relation mining mainly employ text mining and language processing technique on text data and structured data including web pages, user profiles and corpus of literature. Relational Learning [9] refers to the classification when objects or entities are presented in multiple relations. Semantic Role Labeling is a broadly employed text mining technique, as it allows for the addition of structured semantic information to plain text [11]. [15] applies Natural Language Processing to extract protein-protein relationships from rich-annotated corpus in biomedical domain. [6] proposes a general framework for syntax-based relation mining and achieves high accuracy by experimenting with Support Vector Machine as a supervised approach. [17] applies a clustering-based approach to differentiate latent social dimensions from social network, but does not study about the semantic meaning related to the extracted dimensions. [7] learns semantic relationship in a supervised way, treating links as features and requiring labeled pairs as training data. Since it is difficult to find universal features that are useful in every domain, we employ a different philosophy that requires commonsense background knowledge about the correlation between the observed links and the latent roles of the nodes but no training data. To the best of our knowledge, there is no previous work mining semantic relations solely according to a network with neither annotation text nor labeled relations.

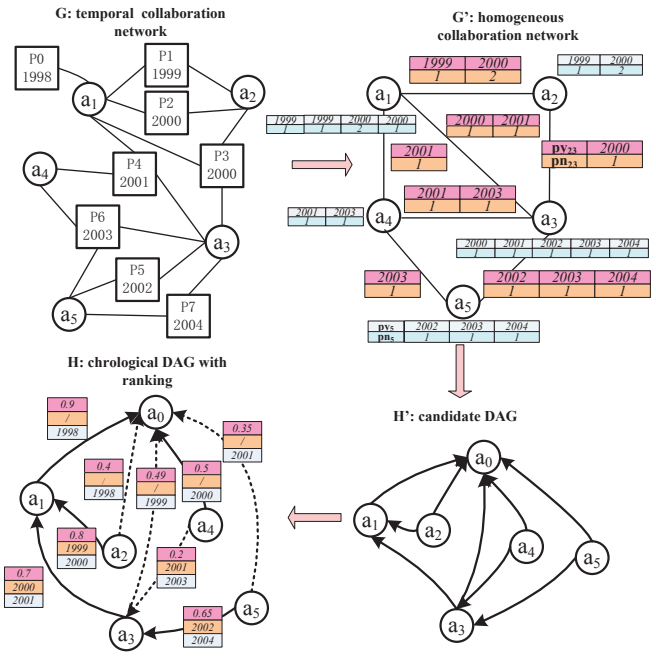



**Figure 2: Example of graph transformation.**

By means of link analysis (*e.g.*, using PageRank[3]), one can compute the importance of a node and the relevance of neighboring nodes. Studies have also shown links can be explored to clean, fuse and reveal the knowledge hidden in a network. For example, *Object Distinction* [20] distinguishes objects with identical names by analyzing their heterogeneous linkages. Furthermore, integrated ranking and clustering can be performed on heterogeneous network based on the link information [16]. [17] proposes to extract latent social dimensions based on network information, and then utilize them as features for discriminative learning. Recognizing the power of links, our approach extracts implicit entity semantic relationships by modeling the network.

To evaluate the discovered advisor-advisee relations we compare them with graduation records maintained by some online projects. Such projects include the Mathematics Genealogy Project [5], the Computer Engineering Academic Genealogy, the AI Genealogy Project and the Software Engineering Academic Genealogy. [19] 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.

## 3. PROBLEM FORMULATION

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

In general, our study takes as input a time-dependent collaboration network $\{G\} = \{(V = V^p \cup V^a, E)\}$, 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. 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 a virtual node $a_0$, which will be the root of an advising tree.). Each edge $e'_{ij} = (i, j) \in E$ connects authors $a_i$ and $a_j$ if and only if they have publication together, and there are two vectors associated with the edge, Pub_Year_vector $\mathbf{py}_{ij}$ and Pub_Num_vector $\mathbf{pn}_{ij}$. They are of equivalent length, indicating the year they have publications and the number of coauthored papers they have at that time. For example, $\mathbf{py}_{ij} = (1999, 2000, 2001), \mathbf{pn}_{ij} = (2, 3, 4)$ indicates that author $a_i$ and $a_j$ have coauthored 2, 3 and 4 papers in 1999, 2000, and 2001 respectively. Similarly, we associate with each author two vectors $\mathbf{py_i}$ and $\mathbf{pn_i}$ to respectively represent the number of papers and the corresponding published year by author $a_i$. The two vectors $\mathbf{py_i}$ and $\mathbf{pn_i}$ can be derived from $\mathbf{py_{ij}}$ and $\mathbf{pn_{ij}}$.

We denote the author $a_i$'s advisor as $a_{y_i}$, where $y_i$ is an introduced hidden variable. If $a_i$'s advisor is $a_j$, we use $[st_{ij}, ed_{ij}]$ to represent the time interval the advising relation lasts. For brevity we denote $st_i = st_{iy_i}$ and $ed_i = ed_{iy_i}$. 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-advisee relationship, we need not only to decide the value of the hidden variable $y_i$ for each author $a_i$, but also to estimate the start and the end years $st_{iy_i}, ed_{iy_i}$. In reality, this problem is more complicated: (i) one could have multiple advisors like master advisors, PhD co-advisors, post-doctorial advisors; (ii) some mentors from industry behave similarly as academic advisors if only judged by the collaboration history; and (iii) one's advisor could be missing in the data set. Therefore, instead of using a boolean model, we adopt a probabilistic model to rank the likelihood of potential advisor(s) for each author. Formally, we denote $r_{ij}$ as the probability of $a_j$ being the advisor of $a_i$. To reduce the number of authors being ranked, it is beneficial to keep only those potential pairs of advisor-advisee. We construct a subgraph $H' \subset G'$ by removing some edges from $G'$ and make the remaining edges directed from advisee to potential advisor. 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 $a_i$ is denoted $Y_i = \{j | e_{ij} \in E'_s\}$, *e.g.*, $Y_3 = \{0, 1\}$. Correspondingly, the index set of potential advisees is denoted $Y_i^{-1} = \{j | e_{ji} \in E'_s\}$.

Then the task becomes finding $r_{ij}, st_{ij}, ed_{ij}$ for every possible advising pair $(i, j) \in E'_s$. So the output is the DAG $H = (V', E'_s, \{(r_{ij}, st_{ij}, ed_{ij})\}_{(i,j) \in E'_s})$. The transformation process is illustrated in Figure 2. 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 predict 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 determine the parameters. If no training data is provided, we can simply use some empirical values, such as the third quartile of all the ranking scores.

## 4. APPROACH

In this section, we first make basic assumptions as the prerequisite of our approach, then propose a two-stage framework and present the approach for each stage. The main idea is to leverage a time-constrained probabilistic factor graph model to decompose the joint probability of the unknown advisor of every author. The time-related information associated 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. One can apply general algorithms for inference on factor graph, e.g., sum-product and JunctionTree. However, these algorithms suffer from the problem of low efficiency. Thus a new message passing algorithm on the candidate graph is designed that approximates the computation and greatly improves the efficiency.

## 4.1 Assumptions and Framework

Commonsense knowledge is needed for recognizing interesting semantic relationships. Here we make a few general assumptions based on the commonsense knowledge about advisor-advisee relationships.

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

This formula reflects the following fact for general consideration of advising relationship. At each time $t$ during the publication history of a node $x$, $x$ is either being advised or not being advised. Once $x$ starts to advise another node, it will never be advised again. $x$ cannot advise $y$ at the year $t_1$ if $x$ is advised by any node $p$ at the year $t_1$. If $x$ advises $y$, the time $y$ is advised by $x$ 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 4.3, we will use this assumption in our model.

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

That means for a given pair of advisor and advisee, the advisor always has a longer publication history than the advisee. $py^1_x$ 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. Due to the order, the candidate graph $H'$ is assured to be a DAG. We will use this assumption in the filtering process in Section 4.2.

Additional assumptions about the correlation between the potential relationship and the publication history will be discussed 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 $H'$. This includes the transformation from $G$ to a 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 of preprocessing is to generate the candidate graph $H'$ and reduce the search space while keeping the real advisor 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 a filtering process is performed 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, the following conditions are checked. First, Assumption 2 is checked. Only if $a_j$ started to publish earlier than $a_i$, the possibility is considered. Second, some heuristic rules are applied, which are based on the prior intuitive knowledge about advisor-advisee relations. Many rules are reasonable but for each there is counter example in real world. It is unknown how well they work before the results are tested. 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 collaborators, $kulc$ (*i.e.*, Kulczinski measure [18] and $IR$ (*i.e.*, imbalance ratio). They are defined as

$$kulc^t_{ij} = \frac{\sum_{py^k_{ij} \leq t} pn^k_{ij}}{2}\left(\frac{1}{\sum_{py^k_i \leq t} pn^k_i} + \frac{1}{\sum_{py^k_j \leq t} pn^k_j}\right) \quad (1)$$

$$IR^t_{ij} = \frac{\sum_{py^k_j \leq t} pn^k_j - \sum_{py^k_i \leq t} pn^k_i}{\sum_{py^k_i \leq t} pn^k_i + \sum_{py^k_j \leq t} pn^k_j - \sum_{py^k_{ij} \leq t} pn^k_{ij}} \quad (2)$$

The Kulczynski measure reflects the correlation of the two authors' publications. [18] 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 an increase in the sequence $\{kulc^t_{ij}\}_t$. For IR, we calculate the sequences in the same way. IR [18] 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 publications than the advisee during the advising time. Then we have the following rule.

Author $a_j$ is not considered to be $a_i$'s advisor if one of the following conditions holds:

$R1$: $IR^t_{ij} < 0$ in the sequence $\{IR^t_{ij}\}_t$ during the collaboration period of $a_i$ and $a_j$,

$R2$: there is no increase in the sequence $\{kulc^t_{ij}\}_t$ during the collaboration period,

$R3$: the collaboration period of $a_i$ and $a_j$ lasts only for one year,

$R4$: $py^1_j + 2 > py^1_{ij}$,

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 methods as YEAR1 and YEAR2. And we refer to YEAR as taking the earlier time of the two years estimated by them. After estimating $st_{ij}$ and $ed_{ij}$, we calculate the average of Kulczynski and IR measure during that period, and use 1)Kulczynski ; 2)IR; 3)the average of the two as three different definitions of the local likelihood. The last definition is formally

$$l_{ij} = \frac{\sum_{st_{ij} \leq t \leq ed_{ij}} (kulc^t_{ij} + IR^t_{ij})}{2(ed_{ij} - st_{ij} + 1)} \quad (3)$$

And the other two are similar. 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: TPFG 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 relationship among individual links. Now we define the TPFG model.

For each node $a_i$, there are three variables to decide: $y_i, st_i$, and $ed_i$. 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.

$$P(\{y_i, st_i, ed_i\}_{a_i \in V^a}) = \frac{1}{Z} \prod_{a_i \in V^a} g(y_i, st_i, ed_i) \quad (4)$$

with

$$\forall a_i \in V^a, ed_{y_i} < st_i < ed_i \quad (5)$$

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

Eq. (5) is the constraint according to Assumption 1. To find the 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 the 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 the first simplification as follows. Suppose $a_i$ and his advisor $y_i$ are 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.*,

$$\{st_i, ed_i\} = arg \max_{st_i < ed_i} g(y_i, st_i, ed_i) \quad (6)$$

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. (5) into the feature function, the objective function becomes

$$P(y_1, \ldots, y_{n_a}) = \frac{1}{Z} \prod_{i=1}^{n_a} f_i(y_i | \{y_x | x \in Y_i^{-1}\}) \quad (7)$$

with

$$f_i(y_i = j | \{y_x | x \in Y_i^{-1}\}) = g(y_i, st_{ij}, ed_{ij}) \prod_{x \in Y_i^{-1}} I(y_x \neq i \vee ed_{ij} < st_{xi}) \quad (8)$$

where

$$I(y_x \neq i \vee ed_{ij} < st_{xi}) = \begin{cases} 1 & y_x \neq i \vee ed_{ij} < st_{xi} \\ 0 & y_x = i \wedge ed_{ij} >= st_{xi} \end{cases} \quad (9)$$

is the identity function. 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. (5) for all hidden variables are decomposed to many local identity function. Now we only need to optimize Eq. (7). Furthermore, to obtain the rank score of each advising relationship, *e.g.*, $a_j$ advise $a_i$ (shortly $a_j \rightarrow a_i$), we can compute the conditional maximal probability

$$r_{ij} = \max P(y_1, \ldots, y_{n_a} | y_i = j) \quad (10)$$

This simplification assures that for each configuration of $\{y_i\}$, the solution achieves either 0 or the conditional optimum given that configuration. 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.

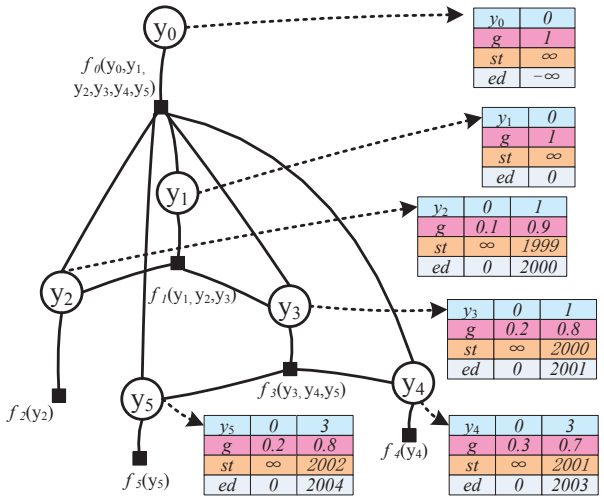



**Figure 3: Graphical representation of a time-constrained probabilistic factor graph, where $\{y_0, \ldots, y_5\}$ are hidden variables defined on all nodes; $f_i(.)$ represents a factor function defined on a hidden variable and its potential advisee sets as neighbors.**

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 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 an 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 in 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 TPFG model. For different kind of relationships, the constraint can vary according to primary assumptions.

## 4.4 Model Learning

To maximize the objective function and compute the ranking score along with each edge in the candidate graph $H'$, we need to infer the marginal maximal joint probability on TPFG, according to Eq. (10). 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* [12] to compute marginal function on a factor graph based on message passing. It performs exact inference on a factor graph without cycles. In the sum-product algorithm, the 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 TPFG is tree-structured factor graph, the message passing rule will be:

$$m_{y_i \to f_j()}(y_i) = \sum_{j' \in Y_i \cup \{i\}, j' \neq j} m_{f_{j'}() \to y_i}(y_i) \quad (11)$$

$$m_{f_j() \to y_i}(y_i) = \max_{\sim\{y_i\}} (\log f_j(y_i, \{y_{i'}\}) + + \sum_{i' \in Y_j^{-1} \cup \{j\}, i' \neq i} m_{y_{i'} \to f_j()}(y_{i'})) \quad (12)$$

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 $y_i$.

Unfortunately, TPFG contains cycles. This algorithm cannot be applied directly. One solution to generalize it is a procedure known as *junction tree algorithm* [2] for exact inference. The junction tree is a tree-structured undirected graph generated from arbitrary triangulated dependency graph, and can be solved by sum-product. Nevertheless, the computational cost of the algorithm is determined by the number of variables in the largest clique and will grow exponentially with this number in the case of discrete variables. 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 6000 variables, not to mention our TPFG has the scale of more than 600,000 variables.

To reduce the computational cost, we can do approximate inference instead of exact inference. A general method *loopy belief propagation* (LBP) [8] simply applies the sum-product algorithm in a cycle-containing graph. It passes message iteratively with flooding schedule. 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 TPFG.

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

The message is passed in a two-phase schema. In the first phase, messages are passed from advisees to possible advisors, and in the second, messages are passed back from advisors to possible 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: $m_{y_i \to f_i()}, m_{f_j() \to y_i}, j \in Y_i$, each of which are along the reverse direction on the edge as in phase 1. The messages are calculated as follows, derived from Eqs. (12) and (11).

$$m_{f_i() \to y_i}(x) = \max_{st_{ki} > ed_{ix}, \forall y_k = i} (\log l_{ix} + + \sum_{k' \in Y_i^{-1}} m_{y_{k'} \to f_i()}(y_{k'})) \quad (13)$$

$$m_{y_i \to f_j()}(x) = m_{f_i() \to y_i}(x) \quad (14)$$

$$m_{y_i \to f_i()}(x) = \sum_{j \in Y_i} m_{f_j() \to y_i}(x) \quad (15)$$

$$m_{f_j() \to y_i}(x) = \max_{st_{kj} > ed_{jy_j}, \forall y_k = j} (\log l_{jy_j} + m_{y_j \to f_j()}(y_j) + \sum_{k' \in Y_j^{-1}, k' \neq i} m_{y_{k'} \to f_j()}(y_{k'})) \quad (16)$$

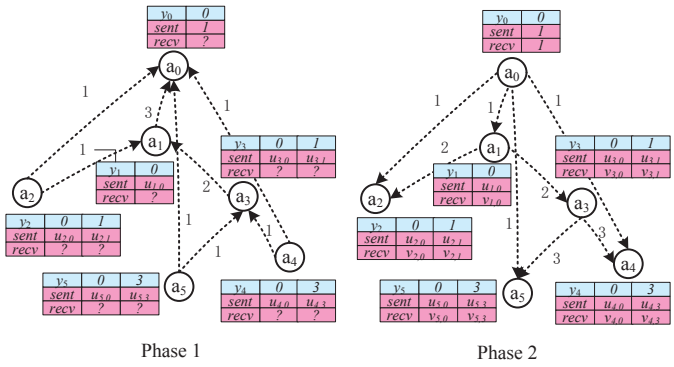



**Figure 4: The 2-phase message passing schema.**

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

$$r_{ij} = \max P(y_1, \ldots, y_{n_a} | y_i = j) = \exp\left(m_{f_i() \to y_i}(j) + m_{y_i \to f_i()}(j)\right) \quad (17)$$

This algorithm still has redundant storage and 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. We further simplify the message propagation by eliminating the function nodes and the internal messages between a function node and a variable node, and we find it equivalent to a message passing problem on the homogeneous graph $H'$. Messages can be seen as being propagated between authors, and the messages can be stored with each author in two vectors: one sent and one received. The order of messages passed is illustrated by the number on each edge in Figure 4. 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 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 derived rules are as follows.

$$\mathbf{sent}_{ij} = \log l_{ij} + \sum_{k \in Y_i^{-1}} \max_{st_{kx} > ed_{ij} \text{ or } x \neq i} \mathbf{sent}_{kx} \quad (18)$$

$$\mathbf{recv}_{ij} = \max_{j' \in Y_j, ed_{jj'} < st_{ij}} (\mathbf{recv}_{jj'} + \log l_{jj'} + + \sum_{k \in Y_j^{-1}, k \neq i} \max_{x \in Y_k, st_{kx} > ed_{jj'} \text{ or } x \neq j} \mathbf{sent}_{kx}) + \sum_{x \in Y_i, x \neq j} \max_{j' \in Y_x} (\mathbf{recv}_{xj'} + + \sum_{k \in Y_x^{-1}, k \neq i} \max_{x' \in Y_k, st_{kx'} > ed_{xj'} \text{ or } x' \neq x} \mathbf{sent}_{kx'}) \quad (19)$$

$$r_{ij} = \exp(\mathbf{sent}_{ij} + \mathbf{recv}_{ij}) \quad (20)$$

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 $O(\sum_{i=1}^{n_a} d_i d'_i)$,

**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. (18);
    $count_j --$;
    **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. (19) and prepare to compute $\mathbf{recv}_{ij}$;
    **end**
  **end**
  **foreach** $i \in Y_(^{-1}j)$ **do**
    Compute $\mathbf{recv}_{ij}$ according to Eq. (19);
  **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.

where $d_i$ and $d'_i$ are the in-degree and out-degree of each node $a_i$ on graph $H'$, respectively. As long as if $H'$ is sufficiently sparse, the maximal degree of the node can be seen as constant $C$ and the complexity is further reduced to $O(n_a)$.

# 5. EXPERIMENTAL RESULTS

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

## 5.1 Experiment Setup

**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. It consists of 654,628 authors and 1,076,946 publications with time provided (from 1970 to 2008). To test the accuracy of the discovered advisor-advisee relationships, we adopt three data sets: One is manually labeled by looking into the home page of the advisors, and the other two are crawled from the Mathematics Genealogy project[1] and AI Genealogy project[2]. We refer to them as MAN, MathGP and AIGP respectively. They only paretically cover the authors in DBLP. We further separate MAN into three sub data sets: Teacher, PhD and Colleague. Teacher contains all kinds of advisor-advisee pairs, while PhD only contains graduated PhDs pairing with their advisors. Colleague contains colleague pairs which are negative samples for advisor-advisee relationship. And we use these data to generate random data sets for test. See Table 1 for details.

**Method.** We compare the proposed TPFG with the following baseline methods:

- Sum-Product+Junction Tree (JuncT). It computes the exact joint probability as the ranking score.
- Loopy Belief Propagation (LBP). It employs an approximate algorithm for inference.
- Independent Maxima (IndMAX). It computes the maximal local likelihood for each variable independently.
- SVM. It is a supervised approach and requires labeled pairs, both positive and negative, as training data.
- RULE. For each author, from all the collaborators that satisfy Assumption 2, choose the one with most coauthored papers.

**Evaluation Aspects.** To quantitatively evaluate our method, we consider two performance measurements: accuracy and scalability. For accuracy, ROC curve is used to evaluate the overall ranking of each prediction, to see whether real advisor-advisee pairs rank higher than non advisor-advisee pairs; and $P@k, \theta$ is used to evaluate the prediction for each individual's advisor, to see whether real advisor ranks on top among all collaborators. We also list a few examples to demonstrate how discovered advisor-advisee relationships can benefit other applications.

The preprocess is implemented with MATLAB 2009a and all experiments are performed on a Desktop 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 [13]. We implement TPFG with Visual C++ 2008. And we use LIBSVM [4] to perform SVM training and prediction.

## 5.2 Accuracy

We conduct a series of experiments to explore the capability of TPFG algorithm in mining advisor-advisee relationships. First, as we mentioned in Section 4.1 and Section 4.2, different assumptions about advising relationships are tested to find the best combination that reflects the reality. Second, we extract small fractions of the whole DBLP network and feed them to TPFG, to prove that the power of network boosts the estimation of joint probability. Finally, we compare our unsupervised approach with a supervised approach. We also tested whether TPFG can be further improved by utilizing training data.

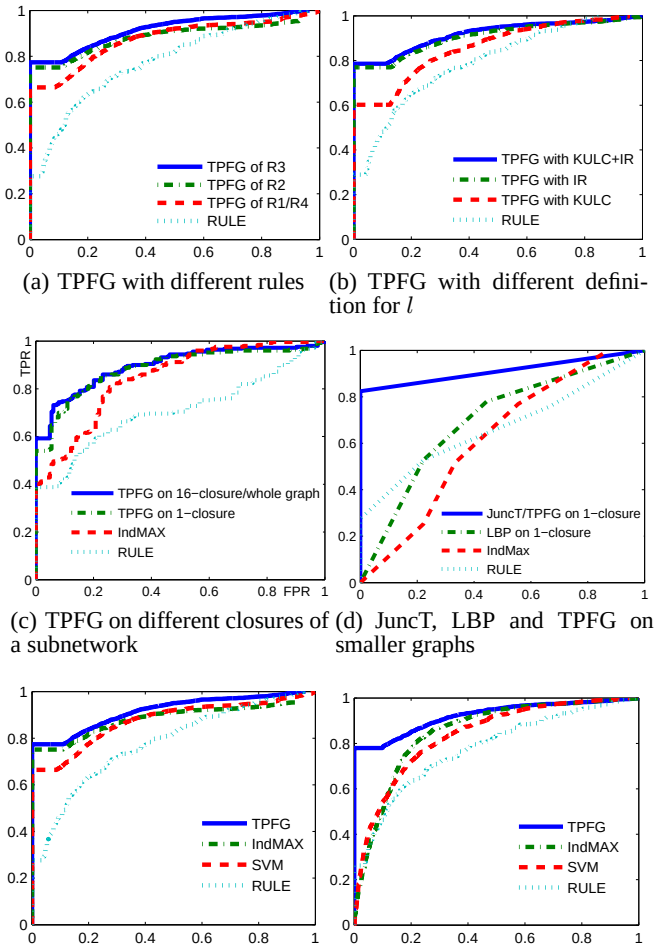
### *5.2.1 Effect of rules in TPFG*

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

The accuracy is compared through ROC curves. For each pair in the tested data, we retrieve the ranking score from the output. Then we sort these ranking score in a descendant order, and plot the ROC curve.

From Figure 5(a) we can see that R2/R3 has the highest suitability on the tested data. R1 and R4 both lead to a slightly worse curve and their curves overlap. In this way we can further refine other rules, including the definition of local likelihood, as shown in Figure 5(b), and estimation of the graduation year, which we found does not affect the ROC curve. In general, by applying a small set of rules our method can extract meaningful knowledge, while which rules to select is flexible depending on the problem specification. It is also observed that TPFG is not sensitive to those rules. For example, if we choose R2, or even R1/R4 other than R3, the worst AOC value 0.88 is not degraded drastically from the optimal choice 0.91. It indicates that our network modeling approach is robust in handling inaccurate local features. From now on we use 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.

[1] `http://www.genealogy.math.ndsu.nodak.edu/`
[2] `http://aigp.eecs.umich.edu/`




(a) TPFG with different rules (b) TPFG with different definition for $l$

(c) TPFG on different closures of a subnetwork (d) JuncT, LBP and TPFG on smaller graphs

(e) IndMAX, TPFG and SVM on TEST1 (f) IndMAX, TPFG and SVM on TEST2

**Figure 5: ROC curves for advisor-advisee prediction**

### 5.2.2 *Effect of network structure*

Using DFS with a bounded maximal depth $d$ from the given set of nodes, denoted as DFS-$d$, we can obtain closures with controlled depth for a given set of authors to test. When $d$ increases, the subnetwork grows larger until it is already the complete closure, *i.e.*, the maximal connected subgraph of $H$ containing the given set. We run TPFG on these closures and plot the ROC curves.

From Figure 5(c) we see that for closures with different depths, TPFG achieves better accuracy when the depth increases, and they all outperform the IndMAX method by more than 5% in AOC. And on the complete closure TPFG reaches the same accuracy as on the whole network since disconnected components will not affect each other.

On these various scaled subnetworks, TPFG achieves different level of approximations to the optimal global joint probability on the whole network. To compare it with the exact maximal joint probability and other approximate algorithm, we show the result on a small graph due to limitations of JuncT and LBP (see Section 4.4). The small graph is constructed by extracting the nodes in PhD and their advisors, and then building 1-closure of it. It consists of 1310 nodes. From Figure 5(d) we find that in the small graph TPFG approximates well to the exact inference algorithm JuncT(AOC difference < 0.01), and outperforms LBP by 16.9%.

**Table 1: Accuracy of prediction by P@(2,$\theta$): $\frac{T}{T+F}$**

| data set | RULE | SVM | IndMAX | | TPFG | |
|---|---|---|---|---|---|---|
| TEST1 | 69.9% | 73.4% | 75.2% | 78.9% | 80.2% | 84.4% |
| TEST2 | 69.8% | 74.6% | 74.6% | 79.0% | 81.5% | 84.3% |
| TEST3 | 80.6% | 86.7% | 83.1% | 90.9% | 88.8% | 91.3% |

TRAIN1=Colleague(491)+PHD(100)
TEST1=Teacher(257)+MathGP(1909)+Colleague(2166)
TRAIN2=TRAIN3=Teacher(257)+Colleague(2166)
TEST2=PHD(100)+MathGP(1909)+Colleague(4351)
TEST3=AIGP(666)+Colleague(459)
IndMAX,TPFG: left - $\theta = 3$rd quartile of $\{r_{ij}\}$; right - trained

### 5.2.3 *Effect of training data*

Support Vector Machines(SVMs) are accurate supervised learning approaches and shown to be successful in syntax-based relation mining[6]. If we treat advisor-advisee pairs as positive examples and non advisor-advisee pairs as negative examples, we can reduce advisor mining to a classification problem 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. For fair comparison with our probabilistic model, we combined Kulczynski and IR measures with what were used in [19] as features.

Direct application of SVM only shows whether a given pair is an advisor-advisee pair, and it is often the case an author is predicted to have multiple advisors, 1001 out of 2657 for TEST1, for example. Thus we examine the probabilistic scores in the test data, and rank them to draw the ROC curve. TPFG is $4.2\%$ and $2.4\%$ higher in AOC than SVM in TEST1 and TEST2 respectively.

Although in this work we define our model as an unsupervised learning approach, it can also work with supervised learning. We have utilized labeled data to select rules in Section 5.2.1. We can also optimize the parameter $\theta$ in the $P@k, \theta$ as we mentioned in Section 3 according to certain criteria such as achieving best information gain on the training data. Then we use the trained parameters to do predictions on test data. Table 1 shows the improvement by utilizing the training data. After training, TPFG can reach an accuracy of $84\%$ to $91\%$.

SVM actually makes a supervised combination of all the assumptions and rules used in TPFG. The difference lies in that it does not explore the constraint and dependency replying on the whole network structure. It does a fairly good job, but still 5-10% worse than optimized TPFG. In conclusion, TPFG can achieve comparable or even better accuracy compared with a supervised method. When parameters are adjusted with training data, its accuracy can be further improved by around $3\%$.

### 5.2.4 *Case study*

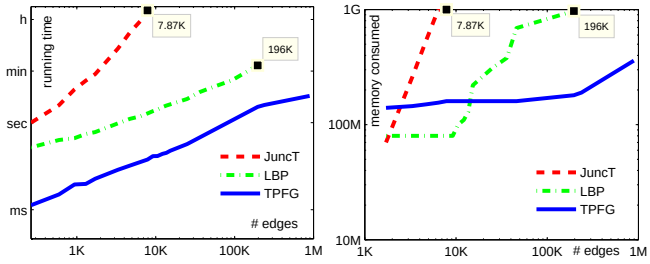
With case study, we find that TPFG can discover some interesting relations beyond the "ground truth" from single source. Table 2 shows some examples. Our ranking results provided with advising time facilitate finding such kind of advising relations, which cannot be easily discovered by referring to Genealogy projects. The mean of deviation of estimated graduation time to the labeled time on the test data sets is $1.76 \sim 1.78$.

We find that at least 40% of the error is contributed by name ambiguity. For example, if we try to find the advisor for "Joseph Hellerstein", our algorithm returns wrong results. If we distinguish "Joseph M. Hellerstein" and his publications properly, our algorithm is able to find the "half" right answer Michael Stonebraker ranked top 1. The answer is half right because there is a co-advisor

**Table 2: Examples of mined relations. Time - the estimated advising time; Note - the factual relation and graduation year**

| Advisee | Top Ranked Advisor | Time | Note |
|---|---|---|---|
| David M. Blei | 1. Michael I. Jordan | 2001-2003 | PhD advisor, 2004 |
| | 2. John D. Lafferty | 2005-2006 | Postdoc, 2006 |
| Hong Cheng | 1. Qiang Yang | 2002-2003 | MS advisor, 2003 |
| | 2. Jiawei Han | 2004-2008 | PhD advsior, 2008 |
| Sergey Brin | 1. Rajeev Motwani | 1997-1998 | "Unofficial advisor"[3] |

[3] cited from a blog of Sergey Brin, who left Stanford to found Google around 1998.




(a) Time (b) Space

**Figure 6: Scalability results in log-log scale. JuncT and LBP fail at 10K and 200K respectively due to memory limitation.**

Jeffrey F. Naughton, who is also ranked high in top 15%. Duplication are even more common among Chinese names. Therefore, if the name DISTINCTION problem [20] is solved well, the accuracy of our algorithm can be further improved. Other reasons for false negatives include that one researcher collaborated with multiple advisors or that one coauthored fewer papers with the advisor. The latter case happens more often for older researchers, for whom the publication data are not as complete as nowadays. More examples are provided in Section 5.4. In those situations, it is almost impossible to find credible advisor-advisee relationships merely based on their publication records. Our study can find typical cases but will miss such atypical cases.

## 5.3 Scalability Performance

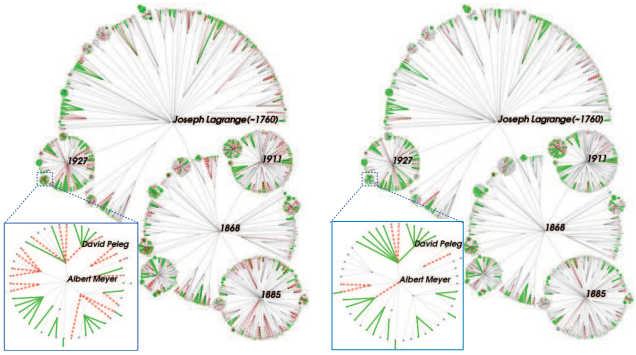
Figure 6 depicts the running time required for different algorithms to infer the probabilistic rank. The same preprocessing is done for IndMAX, JuncT and TPFG, taking 48 minutes. IndMAX and RULE do not perform further learning after preprocessing.

TPFG is shown to be scalable in Figure 6. With regard to the learning time, TPFG costs only 13 seconds on the whole DBLP dataset. As a classification approach, SVM's scalability is related to the size of training data. As an example, the feature computation takes one hour and a half, and the model learning takes 31 seconds for Train1 and 6min26s for Train2.

## 5.4 Applications

The discovered advisor-advisee relationships can benefit many applications, such as online query of advisors, visualization of genealogy, and expert finding, *etc.*. Here we show two examples.

**Visualization of genealogy** The visualized hierarchies of research community based on the relationship can help us gain a better insight of the community. With visualization technique from [10] we can draw the advising tree on hierarchical circles. For example, in Figure 7 we show a subtree of the Mathematics Genealogy. To visualize our results on it, we draw edges with different colors. We have visualized results from both the baseline algorithm RULE and TPFG. The figure is better to be viewed in color mode. We see that

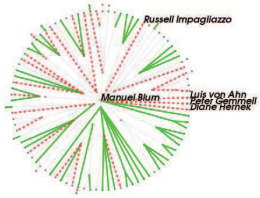



(a) RULE (b) TPFG

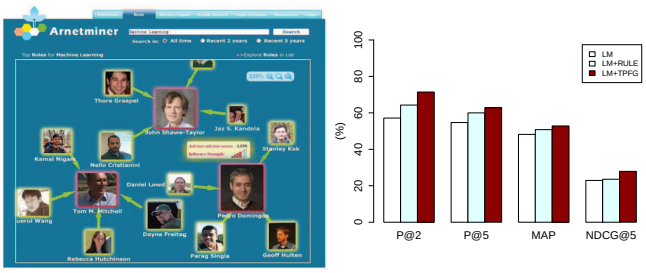
**Figure 7: visualization of advising relationship on genealogy tree, green solid=true positive, red dotted=false negative**

RULE tends to give a ratio close to 1:1 of true positives to false negatives, while TPFG raises this ratio by twice. Zooming into a local region, we see clearly that TPFG is able to identify Prof. David Peleg's students and advisor correctly while RULE could not.

The visualization also leads to some interesting finding. Particularly, in TPFG result, we found the red dotted edges were closer to the root than the green edges. This observation infers that TPFG tends to make mistakes for researchers who graduated earlier. With further statistical analysis, we found that for researchers involved in true positive relations, the average graduate time is 1994 while for false negative results, the average is 1983. We can also analyze collaboration patterns for different research topics and affiliations by looking at those mistakes. For example, the advising tree of some theoretical computer scientists centered in Prof. Manuel Blum has a lot of outliers when detecting their relationship. This implies they have some unusual collaboration patterns. We then found that Diane Hernek and Russell Impagliazzo have no publication coauthored with their advisor, while Peter Gemmell and Luis von Ahn only have 1/8 of their papers with Manuel Blum in DBLP.




**Expert finding and Bole search** Here we illustrate one application on bole search [19], a specific expert finding task, aiming to identify best supervisors (according to their nurture ability [14]) in a specific research field. The task requires advisor-advisee relationships as input which are usually unavailable. To quantitatively evaluate how the advisor-advisee relationships can help bole search, we compare a retrieval method with and without those relationships on a data set used in [19]. Specifically, the data set consists 9 queries (e.g., data mining and machine learning), and for each query, 50 top ranked researchers by ArnetMiner.org are taken as candidates. We sent an email to each of the 50 researchers and another 50 young researchers who start publishing papers only in recent years (>2003) for feedbacks ("yes", or "no", or "not sure"). Finally a list of best supervisors are organized for each query by simply counting the number of "yes"(+1) and "no"(-1) from the 100 received feedbacks. Details can be referred to [19]. For easy comparison, we did not use the learning-to-rank approach (as reported in [19]). Instead, we use the language model (LM), which does not consider the advisor-advisee relationships, and a heuristic-based method which simply combines the language model with




(a) An example of Bole Search (b) Performance

**Figure 8: Application: from Expert Finding to Bole Search**

the advisor-advisee relationships identified by the baseline method (LM+RULE) or identified our proposed approach (LM+TPFG) by

$$s_i = \alpha r_i + (1-\alpha)\frac{1}{|A_i|}\sum_{a_j \in A_i} r_{a_j} \quad (21)$$

where $r_i$ is the relevance score obtained by the language model; $A_i$ is a set of advisees of researcher $a_i$ identified by RULE or TPFG; $\alpha$ is a parameter to trade off the balance between researcher's expertise and his advisees' expertise score. We empirically set $\alpha = 0.7$.

Figure 8 (b) shows the results (Precision@2, Precision@5, mean average precision (MAP), and NDCG@5 [1]) of bole search by the three methods. We see that by considering the advisor-advisee relationships (obtained by either RULE or TPFG), the performance of bole search can be significantly improved. We can also see that with a higher accuracy, our method TPFG clearly achieves a better improvements, particularly for the top two retrieved results (71.4% by TPFG vs. 64.3% by RULE in terms of P@2).

## 6. CONCLUSIONS

We have studied the mining of advisor-advisee relationships from a research publication network as an attempt to discover hidden semantic knowledge in information networks. We propose a two-stage framework to transform a collaboration network step by step until constructing the advising hierarchy with ranking. We propose a Time-constraint Probabilistic Factor Graph (TPFG) model to integrate local intuitive features in the network. Finally, we design an efficient learning algorithm to infer the TPFG model. Experimental results on the DBLP data sets demonstrate the effectiveness of the proposed approach.

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. Another interesting problem is to correlate the discovered latent relationship with social influence analysis. Clearly, much more can be exploited with an information network by exploring inherent knowledge from it.

## 7. ACKNOWLEDGMENTS

Research was sponsored in part by the U.S. National Science Foundation under grant IIS-09-05215, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 (NS-CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. Jie Tang is supported by NSFC (No. 60703059), Key NSFC (No. 60933013), and 863 H&D Program (No. 2009AA01Z138). The authors would like to thank Lu Liu and Yizhou Sun for their help in brainstorming and discussion.

## 8. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

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

[3] 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.

[4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[5] H. B. Coonce. Computer science and the mathematics genealogy project. *SIGACT News*, 35(4):117–117, 2004.

[6] B. Coppola, A. Moschitti, and D. Pighin. Generalized framework for syntax-based relation mining. In *ICDM*, pages 153–162, 2008.

[7] C. P. Diehl, G. Namata, and L. Getoor. Relationship identification for social network discovery. In *AAAI'07*, pages 546–552. AAAI Press, 2007.

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

[9] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[10] Y. Jia and J. C. Hart. Drawing trees: How many circles to use? Technical report, http://hdl.handle.net/2142/14068, 2008.

[11] 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.

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

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

[14] B. K. Mohan. The best nurturers in computer science research. In *SDM*, 2005.

[15] 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.

[16] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD'09*, pages 797–806, 2009.

[17] L. Tang and H. Liu. Relational learning via latent social dimensions. In *KDD*, pages 817–826, 2009.

[18] T. Wu, Y. Chen, and J. Han. Re-examination of interestingness measures in pattern mining: A unified framework. *Data Mining and Knowledge Discovery*, 2010.

[19] Z. Yang, J. Tang, B. Wang, J. Guo, J. Li, and S. Chen. Expert2bole: From expert finding to bole search (demo paper). In *KDD'09*, 2009.

[20] X. Yin, J. Han, and P. Yu. Object distinction: Distinguishing objects with identical names. In *ICDE'2007*, pages 1242–1246, 2007.