TupleRank: Ranking Relational Databases using Random Walks on Extended K-partite Graphs

Jiyang Chen [1], Osmar R. Zaïane [1], Randy Goebel [1], Philip S. Yu [2]

[1] *Department of Computing Science, University of Alberta*
Edmonton, Alberta, Canada T6G 2E8

[1] {jiyang, zaiane, goebel}@cs.ualberta.ca

[2] *Department of Computer Science, University of Illinois at Chicago*
851 S. Morgan St., Rm 1138 SEO, Chicago, IL 60607

[2] psyu@cs.uic.edu

No Institute Given

Abstract. The significant increase in open access digital information has created incredible opportunities for modern database research, especially in exploiting significant computational resources to determine complex relationships within those data. In this paper, we consider the problem of analyzing relational databases and explaining relationships between entities in order to rank tuples based on a notion of relevance. For this purpose, we propose a solution of a class of link analysis algorithms known as the random walk, which can be deployed to discover interesting relationships amongst partial tuples of relational databases that would otherwise be hard to expose. We focus on a shortcoming of the absence of a special kind of relationship, which we call "returning relationship". We demonstrate our ideas on the DBLP database, where we exploit structural variations on relationships between authors, conferences, topics, and co-authorships. We show how a distinction between normal relations and returning relations on objects within that database provides the basis for structuring a random walk algorithm to determine interesting relevance measures. We also show how structural changes in the organization of the random walk can produce novel results that are not attainable with previous database ranking methods.

1 Introduction

Ranking, i.e., a process of positioning entities on an ordinal scale in relation to others, is an important task that has a significant role in many applications. In large databases, users would prefer the top-k partial tuples that are most related to their queries rather than a long list of tuples in a random order. Note that the query result ranking proposed here goes far beyond the functionality of the *ORDER-BY* operator, which can only sort values of specified attributes in numerical or alphabetical order. In social networks, the ordering task is also challenging. One typically measures closeness of related entities in the network

by a relevance score, which is computed with certain similarity metrics based on selected relationships between nodes in a graph. Ranking is also a common notion that has appeared in the context of web search applications. Web pages are ranked and returned as the result of a user query, such that the more relevant the page is to the query, the higher it is ranked. The random walk approach is a popular method for ranking in this context, and has received increasing interest after the undeniable success of the Google search engine, which uses a random walk approach to rank web pages in its search results as well as its list of visited pages to re-index [1]. Simply put, a random walk is a sequence of nodes in a graph such that when moving from one node N to the subsequent one in the sequence, one of N's neighbours is selected at random but with consideration for an edge weight. The relevance of a node, or the importance of a node B with respect to a node A, is the static steady state probability that the sequence of nodes would pass by B when the random walk starts in A. This probability, often estimated by a relevance score, is computed iteratively until some convergence (i.e. when no further changes in the probabilities are observed). A variation of this idea, which we advocate in this paper, is the random walk with restart. Given a graph and a starting point A, if at each step, we select a neighbour of the current point at random proportionally to the edge weights and move to this neighbour, or with probability $P_{restart}$ go back to the initial node A, the sequence of points selected is a random walk with restart (RWR) on this graph. RWR has many applications including neighborhood formulation, automatic image captioning, etc. [2].

In this paper, we are particularly interested in assigning relevance values to categorical data in relational databases. We can then generate useful ranked lists of entities based on this relevance assignment. More specifically, we first construct a database graph based on the relational database, then apply a random walk approach on the graph to obtain rank values for node tuples. Since the random walk algorithm has a start point, the ranking we obtain is only for one specific entity: the starting point in the graph. Here we mainly focus on how to obtain accurate ranking values for one given entity. However, by combining rankings of different starting entities using the methods proposed in [3], we can easily generate a more "global" ranking if necessary, e.g., when there are multiple entities that match the query.

Before applying random walk approaches to compute the relevance score, we need to transform entities in the relational database to a weighted graph which we call the *database graph*. Intuitively, seeing the web as a database, there are two tables, one for page entities and the other for hyperlink relations, which correspond to nodes and edges in the web graph, respectively. Using this observation, we can transform arbitrary relational databases to graphs based on the Entity-Relationship Model (ERM): each entity table is represented as a partition of the graph, where no edges exist within the partition. Each relation table is represented as a set of edges that connect two partitions of the graph. The edge weight is the number of relation tuples that connect two entities in different partitions. (For relations between entities in the same partition, we

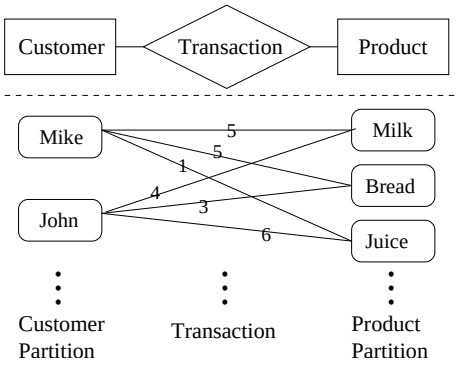

Fig. 1. Create Database Graph based on ERM

can simply create a duplication of the partition.) Figure 1 is an example: the purchasing database of a supermarket can be represented as a bipartite graph, with customers and products as separate partitions and transactions as edges. The edge weight shows how many times one customer bought a product. Note that this has absolutely nothing to do with association rules. Other databases, such as the DBLP bibliography database[1], can also be represented as a bipartite graph, with authors and conferences as graph partitions and paper submissions as edges. These kinds of relations connect one database partition to another, thus we call them *cross-relations*. Unfortunately, on many occasions simple bi-

	Transactions
Mike	{Milk, Bread}(5), {Juice}(1)
John	{Milk}(4), {Bread}(3), {Juice}(6)

Table 1. Product Purchasing Transactions. For example, {Milk,Bread}(5) means Mike bought these two products together five times.

partite representations are deficient in the sense that they cannot capture some important relationships. For example, a bipartite graph can well represent the purchasing relations between customers and products, but the relation that describes how several products are bought in one transaction is missing, e.g., the graph model in Figure 1 fails to show that milk and bread are actually bought together five times by Mike (Table 1). Moreover, if we create a duplicate product partition and use edges between two product partitions to represent these in-the-same-transaction relations, we miss the information of the customer who purchases these products in that transaction, i.e., we capture the fact that milk and bread are bought together five times, but there is no way we can tell the customer is Mike, John or even both. Other simple methods, such as modeling

[1] http://www.informatik.uni-trier.de/ ley/db/

them by adding directional edges, are not practical in the case of a random walk as the approach would hinder the random walk itself and compromise its results. Since these kind of relations describe relations between two nodes A and B in the same partition via one node C in another partition, we call them *returning relations*: the relation from A returns to B via C, which is in a different partition from A and B. Returning relations are common but easy to omit in relational databases. For example, in bibliography data, the co-authorship relation between authors is a returning relation since every co-authorship happened in a particular conference. In a Peer-To-Peer(P2P) environment, the returning relation represents the connection between peers via different seeds. For e-mail senders and recipients, recipients of the same e-mail form a returning relation. Another example comes from microarray data analysis where genes and conditions form a bipartite graph. If we have multiple microarray data sets, the genes activated by a given condition in each microarray data set form a returning relation. Conceptually, the returning relation is a relationship from one partition to itself via another partition, e.g., product to product via customer, author to author via conference, gene to gene via protein or other conditions.

To model returning relations in the database graph, we propose the construction of a virtual layer acting as surrogate nodes to replace the original nodes, then define a random walk algorithm for this extended graph. Our solution is to provide the random walk with effective means to take into account information within these returning relations. Our approach, which is generic to multi-partite graphs with as many intra-partite relations as necessary, consists of two main ideas. One is to introduce the surrogate or virtual nodes to constrain traversal based on the flow control, i.e. the proper connectivity to support a random walk calculation. The other is to capture the returning relations by introducing additional links (including not only reverse links but also special forward links capturing returning relations) to/from the surrogate nodes and assign the proper weights.

Our work has the following contributions:

- We identify novel types of relations in a relational database, model them and extend the bipartite graph model to include various relations between entities. We present an iterative random walk algorithm to compute what we call *relevance scores*, which rank tuples with regard to a given tuple (or node in the database graph). Our ranking scheme considers implicit semantic relationships that are ignored by previous database ranking systems; implicit in the sense that the relationships are not explicitly expressed in the database but inferred by some mechanism such as the random walk.
- We adapt our approach to relational databases with multiple relations and investigate the influence of different random walk directions on a k-partite graph model. The proposed two algorithms can be used in different situations to obtain accurate ranking.
- While ranking results are very difficult to compare and evaluate, we use historical data as ground truth to evaluate our rankings by means of a rec-

ommender system. The idea of using recommendation accuracy to evaluate ranking is novel.

The rest of the paper is organized as follows. We discuss related work in Section 2. Section 3 introduces database graph for relational databases. Random walk algorithms for computing the relevance ranking are described in Section 4. The ranking result and evaluation are reported in Section 5. Finally the study is concluded in Section 6.

2 Related Work

The problem of ranking tuples in the relational database is closely related to the problem of ranking web pages. In the web domain, iterative algorithms [4, 5] on the web graph have been proposed to compute "importance" scores for web pages. Results show that the use of structure can significantly improve the performance of web search engine. Specifically, Page-Rank [5] learns ranks of web pages, which are N-dimensional vectors, by using an iterated method on the adjacency matrix of the entire web graph. In order to yield more accurate search results, Topic-Sensitive PageRank [6] precomputes a set of biased PageRank vectors, which emphasize the effect of particular representative topic keywords to increase the importance of certain web pages. Those are used to generate query-specific importance scores. Alternatively, SimRank [7] computes a purely structural score that is independent of domain-specific information. The SimRank score is a structure similarity measure between pairs of pages in the web graph with the intuition that two pages are similar if they are related by similar pages. Unfortunately, SimRank is computationally very expensive since it needs to calculate similarities between many pairs of objects. According to the authors, a pruning technique is possible to approximate SimRank by only computing a small part of the object pairs. However, it is very hard to identify the right pairs to compute at the beginning, because the similarity between objects may only be recognized after the score between them is calculated. Similarly, a random walk algorithm is usually applied on graphs that represent data models in different fields after structural context has been taken into consideration. For example, the Mixed Media Graph [2] applies a random walk on cross-modal correlation discovery. He et al. [8] propose a framework named MRBIR using a random walk for content-based image retrieval. Sun et al. [9] detect anomaly data in bipartite graphs using the random walk with restart algorithm. Tong et al. [10] proposed a fast solution for applying random walk with restart on large graphs, to save pre-computation cost and reduce query time.

Ranking or top-k query processing for databases are important in many situations such as joining ranked lists or combining scoring functions [11], and have recently attracted much research attention. Most of the available solutions are in the middleware scenario [12, 13] or in relational database systems (RDBMS) setting [14–16], where ad-hoc top-k aggregate queries can also be supported [17]. While these above mentioned works focus on extending the relational algebra

and query optimization to support ranking as a first-class database construct such as the operator *rank-join*[11], we concentrate on ranking tuples based on the original structure and implicit relations deduced from the data.

The idea of representing a relational database as a graph and ranking the tuples based on relational links has appeared in the context of keyword search in [18–20]. In their respective work, the nodes in the graph are database tuples that are found as query results and the relationships between the nodes are induced by foreign key or other given constraints. The ranking values of tuples are computed based on the path distance and types of the connections between these tuples. The main difference is that we naturally transform entity tables into graph partitions and represent relation by edges between partitions to build the database graph.

A similar random walk approach to ranking on structured data is ObjectRank [3]. In this approach, the database graph construction is heavily dependent on the presence of semi-structured data. The random walk is applied based on the schema graph of the database. The probabilities of authority transfer from one type of tuple to another is given by the database administrator or a domain expert. However, the ranking quality of ObjectRank depends on the authority transfer probability setting between every pair of related tuples, which can be hard to find. Another related application work that applies random walk on relational database for ranking is proposed in [21]. In their work, the authors focus on ranking partial tuples instead of tuples, and the database graph is generated based on queries that connect the tuples. Neither of these two approaches considers returning relations in their random walk ranking.

Here in this paper, we build database graphs based on the original relational structure, extend the random walk approach on bipartite graphs to k-partite graphs, and increase the potential of the random walk by expanding the graph with virtual nodes that materialize the semantics of returning relations to obtain a global ranking of the entities, as explained in the following sections.

3 The Database Graph

As mentioned in the Introduction, we can consider the web as a database over a binary relation between pages. It can be well represented by a bipartite graph, where two partitions contains the page set and edges between partitions represent the hyperlinks that connect corresponding pages. We now generalize this idea to arbitrary databases.

Given a relational database $D = (X, Y, R_{X \leftrightarrow Y})$, where entity table $X = \{x_i | 1 \leq i \leq n\}$ and $Y = \{y_j | 1 \leq j \leq m\}$ (there are n tuples in X and m tuples in Y), $R_{X \leftrightarrow Y}$ is the relation table between X and Y: $r(x_i, y_j) \in R_{X \leftrightarrow Y}$ if x_i and y_j are related, e.g., if x_i represents a customer and y_j represents a product, $r(x_i, y_j)$ could mean that the customer x_i has bought product y_j in a transaction. We refer to $R_{X \leftrightarrow Y}$ as the *cross relation* between entity sets X and Y. These are the inter-entity-set relations.

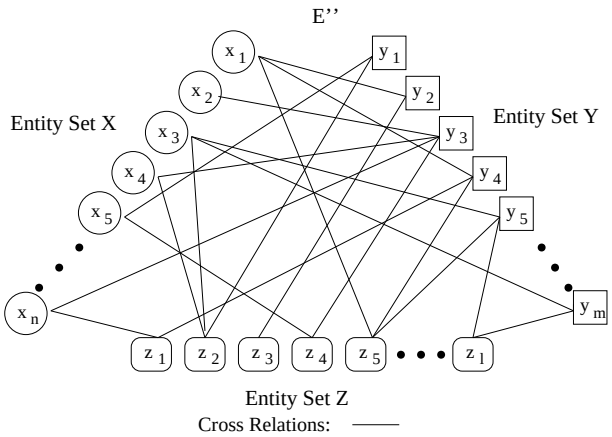

Fig. 2. Tripartite Database Graph for Multiple Cross Relations

3.1 Bipartite Database Graph

Based on the ERM, a relational database $D = (X, Y, R_{X \leftrightarrow Y})$ can be easily transformed to a undirected bipartite graph $G = (X, Y, E)$, where node set $X = \{x_i \mid 1 \leq i \leq n\}$, node set $Y = \{y_j \mid 1 \leq j \leq m\}$ and edge set $E = \{e(x_i, y_j) \mid r(x_i, y_j) \in R_{X \leftrightarrow Y}\}$, i.e., each partite represents an entity set and edges represent relations between entities. We assume the edges are undirected, but they can also be directed for some databases, such as the web. The weight of edge $e(x_i, y_j)$ is $w(x_i, y_j)$, which can be 1 for an unweighted graph, or non-negative value for a weighted graph, where the weight is the number of tuples in the relation table that connect the two entities. Figure 4 (a) shows an example of the bipartite database graph. The bipartite model provides a structure for important information about relations between entities, e.g., nodes in X that are similar to each other usually connect to similar nodes in Y. One can compute the similarity ranking between nodes and the result can be applied to various applications. One possible way to achieve the ranking is to generate the adjacency matrix of graph G and apply a random walk algorithm on that matrix starting from a given node. Similar methods have been applied in [9] to calculate object relevance for anomaly data detection. However, in their approach, they did not take into consideration the returning relations, which is important for object ranking. For example, in the bipartite graph model of customers and products, it may be adequate to only measure cross relations to detect anomalies, but for the problem of similarity ranking between products, how often products are bought together in one transaction, and by which customer must also be taken into account.

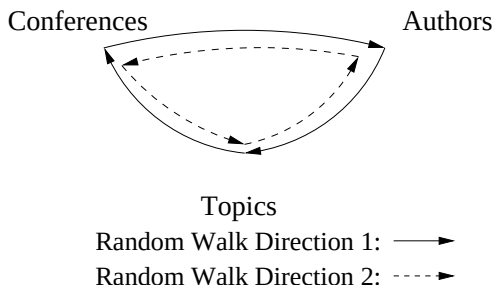

Fig. 3. Random Walks on Tripartite Database Graph

3.2 K-partite Database Graph

We have presented the bipartite database graph for relational database $D = (X \cup Y)$, which has only one type of cross relation, namely $R_{X \hookrightarrow Y}$. However, real world application databases usually have multiple cross relations between several types of entities. For example, there are seeds, peers and file types (multimedia, text, etc.) for P2P system data; conferences, authors and research topics for research publication data; books, readers and categories for a library system; customers, movies, genre and actors for a movie purchase database. We may have more correlated entities in other fields, however, in this paper, we experiment on databases that have cross relations between three types of entities, database graphs for more can be achieved by extending the graph model and applying the proposed method.

We consider a relational database $D' = (X \cup Y \cup Z)$, and we have $R_{X \hookrightarrow Y}$, $R_{Y \hookrightarrow Z}$ and $R_{X \hookrightarrow Z}$ as cross relation tables between these three different types of entities. We naturally use an undirected tripartite graph $G'' = (X, Y, Z, E'')$ to model the data: two entity nodes are connected if they are related. Figure 2 shows the tripartite database graph.

Applying random walk algorithms on a k-partite graph can be interesting, since the direction of the random walk need also to be taken into consideration. For example, assume we have a tripartite database graph representing relations between conferences, authors and topics (Figure 3), there are two possible ways to apply a random walk: assuming we start from authors, walk from author to conference to topic and then back to author, or walk from author to topic to conference and then back to author. Random walk algorithms with both directions rank authors based on the frequency of sharing the same topics and conferences, but there are slight differences due to the walking sequence. Nevertheless, starting from an entity in a particular partition, the random walker need to walk through all related partitions and then return to the beginning partition. By that we no longer need to arbitrarily set the authority transfer probability [3], however the sequence of involved partitions are required for the random walker to jump among different types of entities. We investigate this problem with more details in Section 5.

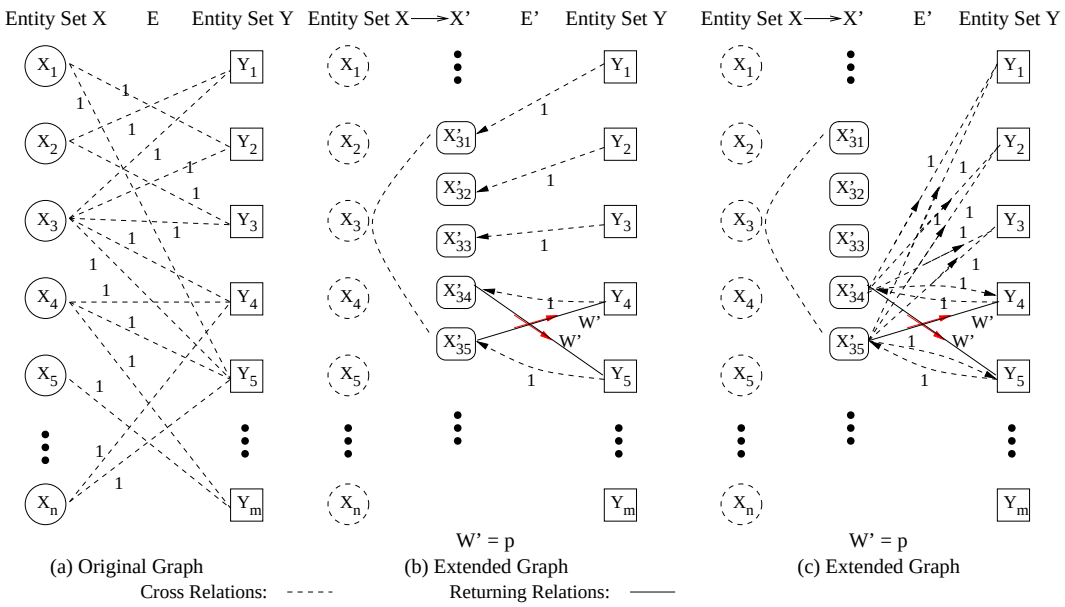

Fig. 4. Database Graphs for Cross Relations and Returning Relations

3.3 Returning Relation Challenges

The aforementioned bipartite and k-partite database graphs are capable of modeling the cross relation $R_{X \leftrightarrow Y}$. However, we may also have another relation set S_Y which they fail to represent: $s(y_i, y_j, x_k) \in S_Y$ if y_i and y_j are related via x_k, e.g., product y_i and y_j are bought together in one transaction by customer x_k or author y_i and y_j have co-authored a paper that is published in conference x_k. We refer to S_Y as the *returning relation* of entity set Y. Returning relations are intra-entity-set relations. For example, in Figure 4 (a), if y_4 and y_5 are related through x_3, i.e., $s(y_4, y_5, x_3) \in S_Y$, there are no edges in the bipartite graph that can be used to represent this returning relation: edge $e(x_3, y_4)$ and $e(x_3, y_5)$ are both used by cross relations between X and Y. More specifically, let us take the customer-product database shown in Table 1 as an example. According to the transactions, Milk, Bread and Juice are all heavily purchased by customers. However, Milk is always bought together with Bread by Mike, therefore, the similarity between Milk and Bread, i.e., the ranking from Milk to Bread and vice versa, should be much higher than other pairs, and that should be obtained through Mike, not any other customer entities.

Several techniques that seemed sensible actually fail to represent returning relations. As we have mentioned, creating a duplicate partition for products and connecting entities which are in the same transactions will not suffice as it does not capture the role of the customer (Figure 5 (a)). In this way, all transactions from different customers are treated equal, but apparently the purchasing frequency of different customers should also be considered in the ranking. Making

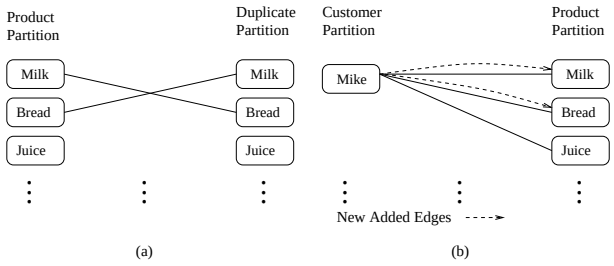

Fig. 5. Attempts to represent returning relations

the link directional and adding reverse links connecting entities involved in returning relations will not work either as one entity can branch to many other nodes in the other partition making the random walk calculation infeasible. For example, after we add two extra links from Milk to Mike and from Mike to Bread, the random walker from entity Milk/Bread does have a higher probability to hit Mike and then visit Bread/Milk, however, it also increases the chance of a random walker from Juice that hit Mike to visit entity Milk and Bread, which is wrong (Figure 5 (b)). Moreover, adding additional nodes to represent each returning relation is impractical when there is a huge number of such relations. For instance, adding "Papers" between Authors and Conferences to make a tri-partite graph would actually not only add a significant number of edges since many authors have multiple papers per conference series, but also, this scheme does not allow the random walk to favour co-athorship as any author or co-author gets the same probability to be visited.

The challenge to capture the returning relation here is not only to provide the connection or path between related nodes, but also control the connectivity or branching from each node so the random walk calculation can be applied while benefiting from the information embedded in the returning relations. A more elaborate database graph is thus required for such a purpose.

3.4 Database Graph for Returning Relations

To solve the problem, we re-structure the bipartite graph by adding surrogate nodes to replace the customer entity nodes and having them linked to both product entity so that random walk calculation can be applied to all intents and purposes.

In more detail, we extend the bipartite graph by adding a virtual level of nodes to replace the partition that do not have returning relations, and add ***direction*** to the edges. Figures 4 (b) and (c) show the details of node x_3 for returning relation $s(y_4, y_5, x_3)$ as an example. (For simplicity, we show only edges related to node $x'_{(3,4)}$ and $x'_{(3,5)}$.) From Figure 4 (a), we know that $r(x_3, y_j) \in R_{X \leftrightarrow Y}$ if $(1 \leq j \leq 5)$, $s(y_4, y_5, x_3)$ can not be expressed since all relevant edges are used in the graph. First, we re-structure the X partition by replacing x_3 with 5 nodes, i.e., the node number equals to the number of nodes that connect

to x_3 (Figure 4 (b)). Since each of the 5 nodes represents the relation between x_3 and y_1, y_2, y_3, y_4, y_5, respectively, the Y nodes connect to their corresponding surrogate nodes with the original weight, e.g. edge $e'(y_4, x'_{3,4})$ (because we had $R_{x_3 \leftrightarrow y_4}$) and $e'(y_5, x'_{3,5})$ (because we had $R_{x_3 \leftrightarrow y_5}$). Then we connect from X' nodes to y_i nodes if y_i has a returning relation with the X and Y node represented by the X' node, the edge is weighted by the returning relation value W', e.g., edge $e'(x'_{3,4}, y_5)$ and $e'(x'_{3,5}, y_4)$. ($W' = p$ for unweighted graphs, which is explained in the following.) We can see in Figure 4 (b) that the returning relationship $s(y_4, y_5, x_3)$, which is missing in the simple bipartite graph, is now represented by extra weight W' of edge $e'(x'_{(3,4)}, y_5)$ and $e'(x'_{(3,5)}, y_4)$.

Although returning relations are well represented in the figure, the new database graph still misses relevant information. For example, y_1, y_2, y_3 are also related to y_4 since they all connect to x_3. However, in the database graph shown in Figure 4 (b), the random walker cannot visit y_1, y_2, y_3 from y_4 via surrogate nodes of x_3. Ideally, the random walker should get the chance to visit all nodes that are related to y_4 and hit the returning-related nodes with higher probability, which is y_5 here. Therefore, we connect from X' nodes to all y_i in Y such that we have $R_{x_3 \leftrightarrow y_i}$. The edge is weighted as the original (i.e., 1), e.g., edge $e'(x'_{3,4}, y_1)$, $e'(x'_{3,4}, y_2)$, $e'(x'_{3,4}, y_3)$, $e'(x'_{3,4}, y_4)$ shown in Figure 4 (c).

Formally, we create an *extended* bipartite graph model $G' = (X', Y, E')$ based on G. We replace each X node x_i into k nodes where $X' = \{x'_{(i,k)} \mid 1 \leq i \leq n, e(x_i, y_k) \in E\}$, i.e., $|X'|$ equals to $\sum_i degree(x_i)$. For edges, at first, each Y node connects to its own representative X' node from different X nodes, i.e. $e'(y_j, x'_{(i,k)}) \in E'$ iff $k = j$, the new weight $w'(y_j, x'_{(i,j)}) = w(x_i, y_j)$. Then, we connect edges from X' to Y: $e'(x'_{(i,k)}, y_j) \in E'$ iff $e(x_i, y_j) \in E$. If $s(y_k, y_j, x_i) \in S_Y$, $w'(x'_{(i,k)}, y_j) = p$, where p is a parameter used to control the returning relation factor. Otherwise, if $s(y_k, y_j, x_i) \notin S_Y$, the weight $w'(x'_{(i,k)}, y_j) = w(x_i, y_j)$. Note that $w'(y_j, x'_{(i,j)}) = w'(x'_{(i,j)}, y_j) = w(x_i, y_j)$ since one can never return to itself via a relation, i.e, $s(y_j, y_j, x_i) \notin S_Y$. For example, one author can never co-author with himself in a conference.

	Publication Records
KDD(K)	A(4), AB(1)
VLDB(V)	AC(3), C(4), E(5)
SIGMOD(S)	B(4), D(1), DE(2)

Table 2. Author Publication Records in Conferences. For example, A, B, C, D, E are authors, $AC(3)$ means that author A and C published three papers together in a certain conference.

To show the necessity and effectiveness of the directed model for returning relations in the database graph, we give a relational database of conferences and authors as an example. Table 2 shows the number of publications of five authors A, B, C, D, E in three conferences KDD VLDB and SIGMOD. Authors A and

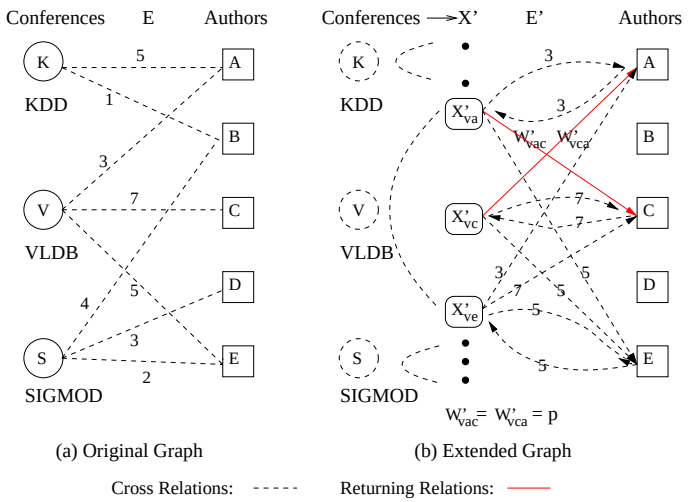

Fig. 6. Directed Model for Returning Relations in Conference-Author Database

C have co-authored 3 papers in VLDB, A and B co-authored 1 paper in KDD and D, E co-authored 2 papers in SIGMOD. In the original database graph, (Figure 6 (a)), author E seems more related to author C since the weights of edges connecting them to VLDB are the heaviest ($W_{VC} = 7$, $W_{VE} = 5$). The influence of the "very" related co-author A is neglected because the model only considers publication frequency. On the other hand, in Figure 6 (b), co-author relations are represented by weights of returning relations ($W'_{vca} = W'_{vac} = p$ where p is a parameter to control the returning relation factor) in the directed model, which shows that author A is more related to C than author E through VLDB due to their collaborations. The parameter p is used to emphasize the co-author influence, we set $p = w * k$ (w is the co-authorship frequency and k is the total author number of a particular conference) so that the random walk probability from conference to authors will not change dramatically for different author numbers. Note that the same extension can be applied on other relations, e.g. co-existing keywords in a paper.

4 Proposed Method

In this section, we first define the relevance score and describe the random walk approach in Section 4.1, then discuss the algorithm to rank the internal and external entities in Section 4.2, finally present the ranking algorithm for entity sets with multiple cross relations in Section 4.3.

4.1 Relevance Score based on Random Walk

Similar to SimRank [7], we believe that two objects are similar to each other if they are related to similar objects for the problem of relevance ranking in the

relational database. Therefore, we consider that an entity is most related to itself and can be assigned a score of 1. We denote the relevance score between entities α and β by $rs(\alpha, \beta)$. We know that $rs(\alpha, \beta) \in [0, 1]$ and $rs(\alpha, \beta) = 1$ iff $\alpha = \beta$. Now the problem of internal and external ranking in the database graph can be described as follows:

Given a node α in partition X of the database graph, we want to compute a relevance score for all nodes $\beta \in X$ (internal ranking) and all nodes γ in other partitions (external ranking). For each partition, the result is a one-column vector containing all relevance scores of the entities with respect to α.

The basic intuition behind our approach is to apply random walks with restart (RWR) from the given node α, and count the visitation frequency that the walk does on each node in the bipartite database graph, i.e., the relevance score of node β is defined as the probability of visiting β via a random walk which starts from α and goes back to α with a probability c. In more detail, RWR in a bipartite graph works as follows: assume we have a random walker that starts from node α. For each step, the walker chooses randomly among the available edges from the current node it stays. After each iteration, it goes back to node α with probability c. The final steady-state probability that the random walker reach node β is the relevance score of β with respect to α: $rs(\alpha, \beta)$. We choose the random walk approach to compute the relevance score because it gives node β high ranking if β and α are connected by many X nodes; this is because the random walker has more paths to reach β from α. The purpose of the periodic restart of the random walk is to raise the chance that close related nodes are visited more often than other nodes.

In the following, we first propose an algorithm to do random walk ranking on a bipartite database graph for returning relations, then present a general algorithm for ranking on k entity sets. These two algorithms can be used in different situations. For example, if users are interested in products that are usually bought together in the same transactions by different customers, algorithm for returning relation can be used; if users want a more general ranking with all related information included, such as product brand, supplier, origin, etc., the general algorithm should be used to compute the ranking.

4.2 Relevance Ranking Algorithm for Returning Relations

At first, we model the entity sets with returning relations as a directed database graph $G' = (X', Y, E')$ (Recall that X has n nodes, Y has m nodes, X' is generated based on X and $R_{X \hookrightarrow Y}$, thus X' has $|E|$ nodes). For better matrix representation, we assume every node in X is replaced by m nodes (x nodes extend with different k ($k \ll m$) in previous discussions). To form the adjacency matrix, we simply put 0 for all rows representing edges that are not in the database graph. Thus, we have $n*m$ nodes in X' and m nodes in Y. For example, to form the adjacency matrix of an unweighted graph as Figure 4 (b), we first

form matrix A for edges from X' to Y:

$$A_{(n*m)\times m} = \begin{pmatrix} ... \\ 1\ 1\ 1\ 1\ 1\ ...\ 0 \\ 1\ 1\ 1\ 1\ 1\ ...\ 0 \\ 1\ 1\ 1\ 1\ 1\ ...\ 0 \\ 1\ 1\ 1\ 1\ p\ ...\ 0 \\ 1\ 1\ 1\ p\ 1\ ...\ 0 \\ ... \end{pmatrix}$$

Then form matrix B for edges from Y to X':

$$B_{m\times(n*m)} = \begin{pmatrix} ... \\ ...\ 1\ 0\ 0\ 0\ 0\ ...\ 0 \\ ...\ 0\ 1\ 0\ 0\ 0\ ...\ 0 \\ ...\ 0\ 0\ 1\ 0\ 0\ ...\ 0 \\ ...\ 0\ 0\ 0\ 1\ 0\ ...\ 0 \\ ...\ 0\ 0\ 0\ 0\ 1\ ...\ 0 \\ ... \end{pmatrix}$$

In matrix A and B, $A(\alpha, \beta)$ or $B(\alpha, \beta)$ denotes that there is a directed edge from node α to node β in G'. If we want to initiate a random walk starting from a node represented by row α in matrix A (the same applies to B), the probability of taking the edge (α, β) is proportional to the edge weight over the weight of all outgoing edges from α. Therefore, we normalize A and B: $P(A) = Norm(A)$, $P(B) = Norm(B)$, such that every row in $P(A)$ and $P(B)$ sum up to 1. We can then construct the adjacency matrix J of G':

$$J = \begin{pmatrix} 0 & P(B)^T \\ P(A)^T & 0 \end{pmatrix}$$

We now transform the given node α into a $(n*m+m)\times 1$ vector $\boldsymbol{v}_\alpha$ initialized with all 0. If $\alpha \in Y$, the vector value for α is set to 1. If $\alpha \in X$, values for all X' node split from α are initialized as $\frac{1}{k}$ (k is the total number of X' nodes out of α). We need to achieve a $(n*m+m)\times 1$ steady-state vector $\boldsymbol{u}_\alpha$, which contains relevance scores over all nodes in the graph model. The steady-state vector can be computed based on the following lemma using the RWR approach.

Lemma 1 *Let c be the probability of restarting random walk from node α. Then the steady-state vector $\boldsymbol{u}_\alpha$ satisfies the following equation:*

$$\boldsymbol{u}_\alpha = (1-c)P(A)\boldsymbol{u}_\alpha + c\boldsymbol{v}_\alpha$$

See [22] for proof.

Algorithm 1 applies the above lemma repeatedly until $\boldsymbol{u}_\alpha$ converges. For all experiments, the restarting probability c is set to 0.15 and converge threshold ϵ is set to 0.1, which give the best performance for RWR according to experiment results in [9]. In step 3, the bipartite structure of the graph model is used to

Algorithm 1 Random Walk Algorithm for Returning Relations

Input: node α, bipartite graph model G, restarting probability c, converge threshold ϵ.

Output: relevance score vector $\boldsymbol{x}$ and $\boldsymbol{y}$ for X and Y nodes.

1. Construct graph model G' for returning relations based on G. Compute the adjacency matrix J of G'.

2. Initialize $\boldsymbol{v}_\alpha = 0$.
If $\alpha \in Y$, set value for α to 1: $\boldsymbol{v}_\alpha(\alpha) = 1$.
If $\alpha \in X$, set values for X' nodes split from α to $\frac{1}{k}$.
(k as defined in text)

3. While ($\Delta \boldsymbol{u}_\alpha > \epsilon$)
$\boldsymbol{u}_\alpha = (1-c)(\frac{P(B)^T \boldsymbol{u}_{\alpha(n*m+1:n*m+m)}}{P(A)^T \boldsymbol{u}_{\alpha(1:n*m)}}) + c\boldsymbol{v}_\alpha$

4. Compute vector $\boldsymbol{x}$ based on $\boldsymbol{u}_{\alpha(1:n*m)}$:
$\boldsymbol{x}(i) = \sum \boldsymbol{u}_{\alpha(1:n*m)}(j)$ if the X' node represented by the jth row is split from node X_i.

5. Set vector $\boldsymbol{y} = \boldsymbol{u}_{\alpha(n*m+1:n*m+m)}$

6. Return $\boldsymbol{x}$, $\boldsymbol{y}$.

save the computation of applying Lemma 1. The result vector $\boldsymbol{u}_{\alpha(1:n*m)}$ and $\boldsymbol{u}_{\alpha(n*m+1:n*m+m)}$ respectively represent vectors of first $n*m$ and last m elements of $\boldsymbol{u}_\alpha$, and contain the relevance score for X' and Y nodes. In step 4, we sum up the scores of X' nodes that are split from the same node to compute relevance scores of original X nodes and directly use $\boldsymbol{u}_{\alpha(n*m+1:n*m+m)}$ as the relevance score vector of Y nodes.

4.3 Algorithm for Multiple Cross Relations

For a relational database with multiple cross relations, we first model the related entities as an undirected k-partite graph. Without loss of generality, we present our algorithm on a tripartite database graph $G'' = (X, Y, Z, E'')$, it can be easily extended to more complex databases. Assume we have n nodes in X, m nodes in Y and l nodes in Z, we can represent all relations using three corresponding matrices: $U_{n\times m}$, $V_{m\times l}$ and $W_{n\times l}$. We normalize them such that every column sum up to 1: $Q(U) = col\_norm(U)$, $Q(U^T) = col\_norm(U^T)$. We then construct the adjacency matrices of G'' after normalization:

$$J_{XY} = \begin{pmatrix} 0 & Q(U) \\ Q(U^T) & 0 \end{pmatrix}$$

$$J_{XZ} = \begin{pmatrix} 0 & Q(W) \\ Q(W^T) & 0 \end{pmatrix}$$

$$J_{YZ} = \begin{pmatrix} 0 & Q(V) \\ Q(V^T) & 0 \end{pmatrix}$$

Similarly, given a node $\alpha \in X$, we want to compute a relevance score for all nodes that are in X, Y, Z. We can transform the given node α into a $(n+m+l)\times 1$

Algorithm 2 Random Walk Algorithm for Multiple Cross Relations

Input: node α, tripartite graph model G'', restarting probability c, converge threshold ϵ.
Output: relevance score vector $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ for X, Y, Z nodes.
1. Compute the adjacency matrices J_{XY}, J_{YZ} and J_{XZ} of G''.
2. Initialize $\boldsymbol{v}_\alpha = 0$, set element for α to 1: $\boldsymbol{v}_\alpha(\alpha) = 1$.
3. While ($\Delta \boldsymbol{u}_\alpha > \epsilon$)
$\boldsymbol{u}_{\alpha(n+1:n+m)} = (Q(U^T) * \boldsymbol{u}_{\alpha(1:n)})$
$\boldsymbol{u}_{\alpha(n+m+1:n+m+l)} = (Q(V^T) * \boldsymbol{u}_{\alpha(n+1:n+m)})$
$\boldsymbol{u}_{\alpha(1:n)} = (Q(W) * \boldsymbol{u}_{\alpha(n+m+1:n+m+l)})$
$\boldsymbol{u}_\alpha = (1-c)\boldsymbol{u}_\alpha + c\boldsymbol{v}_\alpha$
4. Set vector $\boldsymbol{x} = \boldsymbol{u}_{\alpha(1:n)}$, $\boldsymbol{y} = \boldsymbol{u}_{\alpha(n+1:n+m)}$, $\boldsymbol{z} = \boldsymbol{u}_{\alpha(n+m+1:n+m+l)}$.
6. Return $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$.

vector and compute the steady-state vector based on Lemma 1 using the RWR approach on G''. In algorithm 2, the random walk starts from X to Y, then to Z and finally returns to X. The algorithm applies the random walk repeatedly until the vector converges. The result vector contains the relevance scores for X, Y and Z nodes. As mentioned in Section 3.2, there are several possible directions of random walks in a k-partite database graph, which could lead to different ranking results. Algorithm 2 describes the direction of $X \rightarrow Y \rightarrow Z$. The computation sequence and involving matrices in step 3 of the algorithm would be different if the random walk direction is changed. Here we present an algorithm for general database graphs with multiple cross relations only, however, the database graphs can be easily extended to include possible returning relations using the method explained in Section 3.4.

5 Experiments

α	*SIGMOD*	*PODS*	*SAC*	*VLDB*	*DEXA*
1	Database System	Database System (1^{st})	Genetic Algorithm (15^{th})	Database System (1^{st})	Database System (1^{st})
2	Relational Database	Query Language (13^{th})	Neural Network (6^{th})	Relational Database (2^{nd})	Information System (4^{th})
3	Management System	Concurrency Control (17^{th})	Web Service (5^{th})	Information System (4^{th})	Object-Orient Database (8^{th})
4	Information System	Relational Database (2^{nd})	Information Retrieval (12^{th})	Management System (3^{rd})	Expert System ($> 20^{th}$)
5	Web Service	Deductive Database ($> 20^{th}$)	Information System (4^{th})	Data Modeling (10^{th})	Management System (3^{rd})
6	Neural Network	Query Optimization (11^{th})	Data mining (9^{th})	Data Management (7^{th})	Information Retrieval (12^{th})

Table 3. Related Topics for Conference using bipartite model: Conference→Topic→Conference ((x), x is the rank of the topic with respect to SIGMOD

We tested our ranking approach on the DBLP database, the data structure of which is shown in Figure 7. As a relational database, DBLP originally has

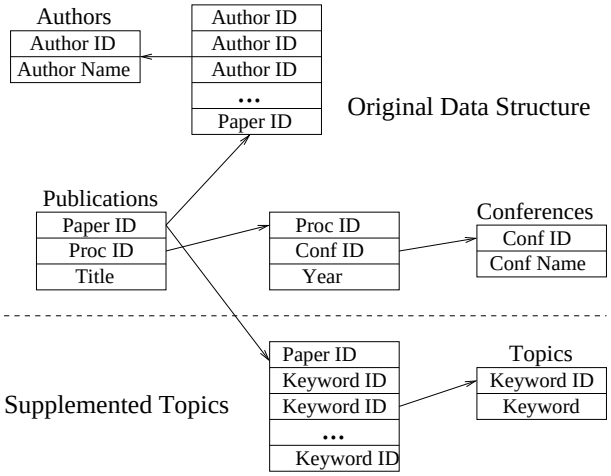

Fig. 7. Our Data Structure extracted from DBLP Database

only two entity types: conference and author. Papers are relations that connect these two kinds of entities. In order to test our approach on k-partite database graphs, we extracted a third type of entities from DBLP data, which is the research topic. Details are explained below.

In the DBLP database, we safely discarded all journal publications since they are only a small fraction compared to the whole database. We also observe that authors in different research areas publish in a certain group of conferences and seldom publish across multiple areas, i.e., the undirected author-conference database graph can be partitioned into several nearly non-overlapping subgraphs, where few edges between authors and conferences connect across partitions. Therefore, it is unnecessary to run the algorithm on the whole graph, instead, we applied a graph partitioning algorithm first and then performed random walks only on the partition containing the given node. There are several algorithms available for graph partitioning. Note that the proposed random walk approach is independent of the selected partitioning algorithm. In our work, we used the METIS algorithm [23] to partition the large graph into 10 subgraphs of about the same size. We examined all our experiments on the biggest partition with 1,170 conferences and 35,926 authors. This partition includes most conferences in the area of Database and Data Mining, e.g., KDD, SIGMOD and VLDB.

Since DBLP data provide paper titles as the only content-related information, we obtained as many paper abstracts as possible from Citeseer [2], then extracted topics based on keyword frequency from both titles and abstracts. At first, we manually selected a list of stop words to remove meaningless but frequent words, e.g., "approach", "using". Then we counted the frequency of every co-located pairs of words (stemmed bi-grams) and selected the top k most frequent bi-grams

[2] http://citeseer.ist.psu.edu/

as our topic (k=1000 in our experiments). We chose to represent topics by bi-grams because most of the research topics can be well described by two words, e.g., artificial intelligence, software engineering and machine learning. Moreover, we added several tri-grams, e.g. Support Vector Machine, World Wide Web if we observed both bi-grams from them (e.g. Support Vector and Vector Machine) to be frequent.

On the performance level, calculating the ranking values in real time is computationally expensive, especially when there are multiple ranking requests. Fortunately, we are able to precompute and save the rankings for any entity in the database. Note that although the graph model can be very large, there are known solutions for a parallel random walk that would apply here [24].

5.1 Ranking for Conference Entities

After we extracted conference-author-topic data from the DBLP database to build the database graph, we checked whether entities with high relevance scores are truly related to the given node. In the following, we select related entities with top 10 relevance scores for given conferences and verify that the result makes sense in the academic context.

Tables 4 and 5 show the top 10 relevance ranked conferences for a given conference α in the bipartite (without topics) and tripartite (with topics) database graph, respectively. We do not make any claim on which list (Table 4 or Table 5) is better, since comparing ranking results itself is still an open problem. Table 4 is based on a conference-author network, therefore, the result implies that the given conference α shares researchers with these conferences who have high relevance scores. One can observe that our ranking puts database conferences as the most relevant conferences for SIGMOD and VLDB. For KDD and ICDM, however, it appears that database conferences are the most relevant. This is due to historical reasons since prominent data mining authors used to and still publish in VLDB, SIGMOD and ICDE venues. The ranking changed in Table 5 since the database graph now includes topics, which increase relevance scores of conferences that share the same research topics with the given conference. For example, after the topic set is taken into account, PODS is no longer top 10 related to SIGMOD. Instead, we see SAC appear in the list due to the similarity of covered research topics. As we can see in Table 3, the most related topics to SIGMOD and VLDB (1^{st} for SIGMOD in Table 5) are almost the same, while SAC (6^{th}) and SIGMOD share several topics as well. On the other hand, PODS (ranked as 13^{th}), which is surprisingly not in the top 10 related conferences for SIGMOD based on a topic-included database graph, does not have many topics in common with SIGMOD (except "relational database" and "database system"). Another example is DEXA, which is ranked high for SIGMOD and VLDB in both tables, we can see that the research interests overlap on quite a few topics in Table 3. As mentioned before, topics in our experiments were extracted from paper titles and abstracts when available. A better topic extraction means could lead to even better rankings.

Table 7 shows the top 10 related authors for SIGMOD and ICDM. All of these authors are researchers in the database and/or data mining area, and have a large number of publications in DBLP with attached topics related to the topics of the conferences in question. Notice that our ranking tends to favor objects center to the social networks. In other words, if an author is highly connected to other authors related to a conference α, and is central in the social network of topics and other conferences related to the conference α, tupleRank would tend to rank this author high vis-à-vis this conference, which justifies why Jiawei Han for instance is ranked the highest for SIGMOD.

α	*KDD*	*ICDM*	*SIGMOD*	*VLDB*
1	VLDB	KDD	VLDB	SIGMOD
2	ICDE	ICDE	ICDE	ICDE
3	NIPS	VLDB	PODS	DEXA
4	ICML	PAKDD	DEXA	PODS
5	SIGMOD	SIGMOD	EDBT	CIKM
6	ICDM	ICML	CIKM	EDBT
7	IJCAI	SDM	KDD	KDD
8	AAAI	IJCAI	DASFAA	BDA
9	PKDD	PKDD	SSDBM	ER
10	CIKM	NIPS	ER	DASFAA

Table 4. Top 10 Related Conferences for Conference using bipartite model: Conference→Author→Conference

α	*KDD*	*ICDM*	*SIGMOD*	*VLDB*
1	VLDB	VLDB	VLDB	ICDE
2	ICDE	ICDE	ICDE	SIGMOD
3	SIGMOD	KDD	DEXA	DEXA
4	NIPS	NIPS	CIKM	SAC
5	DEXA	SIGMOD	DASFAA	DASFAA
6	PAKDD	PAKDD	SAC	CIKM
7	IJCAI	DEXA	ER	ER
8	SAC	IJCAI	IJCAI	IJCAI
9	ICML	SAC	SIGIR	SIGIR
10	SIGIR	ICML	KDD	KDD

Table 5. Top 10 Related Conferences for Conference using tripartite model: Direction Conference→Topic→Author →Conference

Future Collaborator	*Path* (degree of separation)	*Top Topic*	*Top Conf.*
Hans-Peter Kriegel	Philip S. Yu→Jiawei Han→Martin Ester→Hans-Peter Kriegel (3)	Similarity Search	ICDE
Hector Garcia-Molina	Philip S. Yu→Jun Yang→Hector Garcia-Molina (2)	Database System	ICDE
Elisa Bertino	Philip S. Yu→Jiawei Han→Beng Chin Ooi→Elisa Bertino (3)	Data Mining	ICDE
Elke A. Rundensteiner	Philip S. Yu→Yun-Wu Huang→Elke A. Rundensteiner (2)	Data Stream	CIKM
Divyakant Agrawal	Philip S. Yu→Balakrishna R. Iyer→Divyakant Agrawal (2)	Data Stream	VLDB

Table 6. Top 5 Related Author for Philip S. Yu with most recommended Topic and Conference to collaborate ($A \rightarrow B$ means A and B are co-authors)

5.2 Ranking for Author Entities

We ranked various entities (conferences, topics, authors) with respect to a given conference in previous experiments. However, the given entity to start the random walk is not limited to one particular set, it can be any entity across the k-partite database graph. For example, we can also rank related topics, conferences and authors for a given author. Among these possibilities, ranking author for author could provide promising potential collaboration recommendations for researchers, i.e., high relevance score between authors implies that their research topics and fields, which are represented by conferences, are very similar.

We applied our random walk algorithm on the conference-author-topic database graph. We removed all the researchers that have already co-authored a paper with the given author from the ranked list. Therefore, there are no direct collaboration connections between the given author and the listed authors. In Table 6, we list the top 5 authors recommended as possible future collaborators for Philip S. Yu. To validate our results, we show the paths between them, i.e. degree of separation ($A \rightarrow B$ means A and B are co-authors) and their most frequent topic and conference in common. Obviously, all recommended collaborators are strongly connected, only one or two steps away via co-authorship, to Philip S. Yu, and the listed topics and conferences are aparently all relevant to his research interests. A more systematic validation of this kind of recommendation using chronological data from DBLP is given in the next section.

5.3 Evaluating the Returning Relation Model

In the above, we validated the effectiveness of our approach in ranking similar entities in database graphs with cross relations. However, taking returning relations into consideration can further improve the accuracy of the ranking. Table 8 shows the top 10 related researchers (co-author included) for Jiawei Han based on the bipartite database graph without the returning relation and the database graph with the returning relation (RR). The number shown after the name is the number of co-authored papers with Jiawei Han. It is obvious that authors should be most similar, in terms of relevance score, to their co-authors, i.e., collaborators should be ranked higher than other entities. In the table, most of the authors that achieve high ranking in the first database graph never co-authored with Jiawei Han. The reason is that the database graph is built based

α	*SIGMOD*	*ICDM*
1	Jiawei Han	Jiawei Han
2	Hans-Peter Kriegel	Philip S. Yu
3	Michael Stonebraker	Jian Pei
4	Elisa Bertino	Masaru Kitsuregawa
5	David J. DeWitt	Wei Wang
6	Philip S. Yu	Eamonn J. Keogh
7	Georges Gardarin	Hans-Peter Kriegel
8	Elke A. Rundensteiner	Hongjun Lu
9	Michael J. Carey	Ming-Syan Chen
10	Hongjun Lu	Osmar R. Zaïane

Table 7. Top 10 Related Authors for Conference using tripartite model: Direction Conference→Topic→Author →Conference

on conference-author relations only, thus these authors are actually the ones that have the most similar conference publication record as Jiawei Han. On the other hand, in the results of the RR graph, we see that the ranking sequence basically follows the frequency of the co-authorship, i.e., the more frequent an author collaborates with Jiawei Han, the higher ranking s/he has. There are exceptions, e.g., Philip S. Yu is ranked second with 15 collaborated publications, which is only third in collaboration frequency. It happens because the RR graph still considers conference-author relations, and Philip S. Yu is ranked first in the first graph, which means he is most related to Jiawei Han regarding publication at conferences. Therefore, our approach is more sensible to rank the entities by considering not only cross relations between different kinds of entities, but also returning relations between the same kind of entities.

	Bipartite Database Graph	*RR Graph*
1	Philip S. Yu(15)	Jian Pei(22)
2	Hans-Peter Kriegel(0)	Philip S. Yu(15)
3	Christos Faloutsos(0)	Xifeng Yan(19)
4	Rakesh Agrawal(0)	Dong Xin(14)
5	Hector Garcia-Molina(0)	Ke Wang(10)
6	Raghu Ramakrishnan(0)	Wei Wang(6)
7	Carlo Zaniolo(0)	Xiaolei Li(9)
8	Surajit Chaudhuri(0)	Osmar R. Zaïane(9)
9	H. V. Jagadish(0)	Hong Cheng(7)
10	Hongjun Lu(5)	Jianyong Wang(9)

Table 8. Compare Results for Jiawei Han of Original Model and Returning Relation(RR) Model ((*x*), *x* is the number of paper co-authored with Jiawei Han)

Unfortunately, since there is no ground truth for the ranking, it is difficult to evaluate the performance of our ranking approach in comparison with other

rankings. However, in the special context of DBLP, we can use a time window on the publication history to measure the accuracy of our ranking list if TupleRank rankings are used to suggest potential close collaborators. More specifically, for one given author, we compute the top-k ranked authors based on early records. These are the recommended collaborators. Then measured the accuracy by the actual collaborations that were obtained from later historical data. In more detail, we applied our approach on database graphs that are built based on publication records that are prior to the year 2002 and used the rest (2002-2006) of the records to test the ranking list, which contains the top k ranked authors that are not co-authors of the given original author prior to 2002 (k=200 in our experiments). We used the co-author distance or degree of separation η to measure the closeness of any two authors, e.g., $\eta = 1$ for two authors that ever published a paper together, $\eta = 2$ for two researchers if they have a co-author in common, and so on. $\Delta_{\eta(i)}$ represents the change of the distance of a recommended author i between time periods:

$$\Delta_{\eta(i)} = \eta(i)_{2002} - \eta(i)_{2006} >= 0$$

If a recommended researcher is getting closer to the given author, we believe our approach is accurate in ranking highly-related entities. The recommender system is not the purpose of this paper. It is solely used as a means to evaluate our ranking.

Evaluation results for a few authors based on the RR database graph are shown in Table 9, in which the first column shows the number of high-ranked authors who did collaborate with the given author later (i.e. became co-authors between 2002 and 2006). The second column is the number of authors that become closer. The third is the total change of the co-author distance. However, evaluating only based on these numbers is incomplete: it is always easier for people with large η to get closer to the given author and is much harder for strongly connected researchers to achieve shorter distance since they are already very close. Therefore, we evaluate by ψ, which is the average distance reduction over the sum of the distances of these recommended authors who did become closer (prior to 2002).

$$\psi = \frac{\sum \Delta_{\eta(i)}}{\sum \eta(i)_{2002}}$$

The last column of the table shows the ψ value. Surprisingly the ranking performance, measured by ψ, is quite stable for all listed authors. Note that many new co-authors appeared in 2002-2006 but did not exist in the database prior to 2002, thus are not participating in the ranking.

Therefore, we believe that our approach based on the RR database graph not only achieves accurate ranking of co-authors for a given author, but is also excellent in finding related researchers as possible future collaborators. One good example is that David J. DeWitt is ranked 4th for Raghu Ramakrishnan based on data prior to 2002. In fact, they have co-authored five papers after 2002 in ICDE, KDD, VLDB and SIGMOD, according to DBLP.

	$\eta(i) \to 1$	n	$\sum_n \Delta_{\eta(i)}$	ψ
Hans-Peter Kriegel	0	36	39	0.267
Osmar R. Zaïane	0	56	88	0.314
Elisa Bertino	2	22	23	0.323
Jiawei Han	3	21	23	0.343
Jian Pei	6	38	44	0.346
Philip S. Yu	7	45	48	0.347
Rakesh Agrawal	2	6	6	0.352
Raghu Ramakrishnan	7	19	19	0.358

Table 9. Evaluate Ranking by Future Collaboration from Publication History: $\eta(i) \to 1$ means becoming a co-author; n is the total number of authors that get closer; $\sum_n \Delta_{\eta(i)}$ is the total Δ distance; ψ is the average distance reduction.

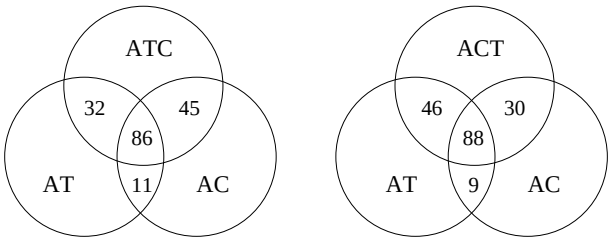

Fig. 8. Random Walks on Tripartite Model

5.4 Random Walk on tripartite Graph

As mentioned in Section 3.2, there are two possible directions of random walks in a tripartite graph, (and C_k^2 options for a k-partite graph), which could lead to different ranking results. To choose the most appropriate direction for a given database graph is an interesting and unsolved problem. We experimented on the conference-author-topic database from DBLP and present our result in the following.

Assume we start from an author node to rank related authors for that author, the two possible random walk directions of the conference-author-topic database are: *author* → *conference* → *topic* → *author* and *author* → *topic* → *conference* → *author* (Figure 3). Each entity set affects the final ranking result based on its cross relations, e.g., relation set conference↔author affects the relevance score result since authors that published in the same conference as the given author would have high ranking. Usually those relation sets are not equal in significance, for example, one may think relations between topics and authors are more important and should be emphasized more in the random walk than other relations. We need to investigate influence changes of different directions to find the appropriate sequence of entity sets that the random walker should follow, based on the importance of these entity sets.

Given author "Jiawei Han", we run the random walk algorithm on the bipartite database graph between authors and conferences (AC), on the bipartite database graph between authors and topics (AT), on the tripartite graph following the direction *author* → *conference* → *topic* → *author* (ACT), and on the tripartite graph following the direction *author* → *topic* → *conference* → *author* (ATC). We selected authors with top k (k=200) relevance score as a test list for the given author. We show the result in Figure 8, where numbers in the overlapping area of the circles represent the number of authors that these ranking lists share. For both directions, most of the authors in the list (81.5% of ATC, 82% of ACT) can be found in either AT or AC, which confirms the initial observation that each entity set has influence on the result of the multiple cross relation model. However, their influences change for different directions. AT contributes 32 authors to ATC and that number increase to 46 for ACT. On the other hand, AC's contribution drops from 45 for ATC to 30 for ACT. The experiment shows that, for author "Jiawei Han", the influence of entity sets on the ranking is increasing following the sequence of the random walk, i.e., topics are more emphasized in ACT and conferences are more important in ATC, as the last partition of the random walk. In order to investigate whether this phenomenon is general, we run the same experiment on more random authors, whose results all show similar characteristics. Therefore, we believe that the random walk direction in a tripartite graph for the multiple cross relation model can affect the relevance ranking result. The later the entity set is visited by the random walk, the more influence it has on result scores and the more important it should be. Similar behaviour is expected for k-partite graphs.

5.5 Discussion

Ranking entities in a relational database, where objects are cross-linked with each other via multi-type links, is important in discovering the rich semantic information these links may contain and in identifying the paramount relationships among objects. Due to limitations of the DBLP database, it is hard to extract accurate topics for conference papers since the only available content-related information are the title of the paper and incomplete abstracts from Citeseer. In experiments where we used only single keywords as topics, we observed that the majority of the entities of ACT or ATC (see Section 5.4) can be found in AT, and the change of direction did not affect much AT's contribution. In other words, inaccurate topic-author/topic-conference relations have an overwhelming influence on the random walk in the tripartite graph model and the result ranking score. Using bi-grams greatly improves the quality of relations and ranking results. Better performance can be expected if the topics are extracted or provided more accurately. However, classifying topics is itself a huge research issue and is out of the scope of this paper.

6 Conclusions

A wide range of databases can be described as related entities sets and can be modeled as k-partite graphs, such as P2P networks, author-conference relationships, customer-movie rental records, etc. This paper addresses two problems for such models: firstly, a random walk algorithm is proposed for relevance score calculation in models with returning relations; secondly, the model and algorithm for multiple cross relations are presented and the consequences of different random walk directions are investigated. These two algorithms can be used in different situations. We validate results of the algorithms on existing publication databases and evaluate our methods by predicting later collaborations based on early publication records. Our experimental results confirm the accuracy and effectiveness of the proposed methods.

References

1. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: WWW, Brisbane, Australia (1998) 107–117
2. Pan, J.Y., Yang, H.J., Faloutsos, C., Duygulu, P.: Automatic multimedia cross-modal correlation discovery. In: KDD. (2004) 653–658
3. Hwang, H., Hristidis, V., Papakonstantinou, Y.: Objectrank: Authority-based keyword search in databases. In: VLDB. (2004) 564–575
4. Kleinberg, J.: Authoritative sources in a hyperlinked environment. In: Proceedings of the Ninth Annual ACM-SAIM Symposium on Discrete Algorithms. (1998)
5. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. In: Technical report, Stanford University Database Group. (1998)
6. Haveliwala, T.H.: Topic-sensitive pagerank. In: WWW. (2002) 517–526
7. Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: KDD. (2002)
8. He, J., Li, M., Zhang, H.J., Tong, H., Zhang, C.: Manifold-ranking based image retrieval. In: MULTIMEDIA: Proceedings of the 12th annual ACM international conference on Multimedia. (2004) 9–16
9. Sun, J., Qu, H., Chakrabarti, D., Faloutsos, C.: Neighborhood formation and anomaly detection in bipartite graphs. In: ICDM. (2005) 418–425
10. Tong, H., Faloutsos, C., Pan, J.Y.: Fast random walk with restart and its applications. In: ICDM. (2006) 613–622
11. Ilyas, I.F., Aref, W.G., Elmagarmid, A.K., Elmongui, H.G., Shah, R., Vitter, J.S.: Adaptive rank-aware query optimization in relational databases. TODS **31**(4) (2006) 1257–1304
12. Bruno, N., Gravano, L., Marian, A.: Evaluating top-k queries over web-accessible databases. In: ICDE. (2002)
13. Chang, K.C.C., won Hwang, S.: Minimal probing: Supporting expensive predicates for top-k queries. In: SIGMOD. (2002)
14. Ilyas, I.F., Aref, W.G., Elmagarmid, A.K.: Supporting top-k join queries in relational databases. In: VLDB. (2003) 754–765
15. Ilyas, I.F., Shah, R., Aref, W.G., Vitter, J.S., Elmagarmid, A.K.: Rank-aware query optimization. In: SIGMOD. (2004) 203–214

16. Li, C., Chang, K.C.C., Ilyas, I.F., Song, S.: Ranksql: query algebra and optimization for relational top-k queries. In: SIGMOD. (2005) 131–142
17. Li, C., Chang, K.C.C., Ilyas, I.F.: Supporting ad-hoc ranking aggregates. In: SIGMOD. (2006) 61–72
18. Aditya, B., Bhalotia, G., Chakrabarti, S., Hulgeri, A., Nakhe, C., Parag, S.S.: Banks: Browsing and keyword searching in relational databases. In: VLDB. (2002) 1083–1086
19. Agrawal, S., Chaudhuri, S., Das, G.: Dbxplorer: A system for keyword-based search over relational databases. In: ICDE. (2002)
20. Hristidis, V., Papakonstantinou, Y.: Discover: keyword search in relational databases. In: VLDB. (2002) 670–681
21. Geerts, F., Mannila, H., Terzi, E.: Relational link-based ranking. In: VLDB. (2004) 552–563
22. Strang, G.: Introduction to linear algebra (Wellesley-Cambridge Press, 3 Edition, 1998)
23. Karypis, G., Kumar, V.: Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distriuted Computing **48**(1) (1998) 96–129
24. Youssef, A.: A parallel algorithm for random walk construction with application to the monte carlo solution of partial differential equations. IEEE Trans. Parallel Distrib. Syst. **4**(3) (1993) 355–360