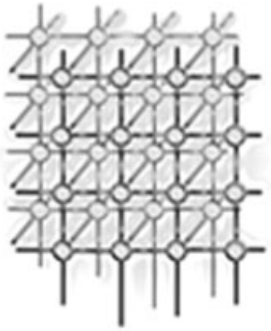
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2007; **19**:2031–2046
Published online 19 December 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1145

Using semantic links to support top-K join queries in peer-to-peer networks



Jie Liu*,†, Liang Feng and Hai Zhuge

Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology,
Chinese Academy of Sciences, Beijing 100080, People's Republic of China

SUMMARY

An important issue raised in peer-to-peer (P2P) applications is how to accurately and efficiently retrieve a set of K best matching data objects from different sources while minimizing the number of objects that have to be accessed. The proposed solution is to organize peers by a semantic link network representing the semantic relationships between peers' data schemas. Queries are only routed to semantically relevant peers. A pruning-based local top-K ranking approach is proposed to reduce the transmitted data by pruning tuples that cannot produce the desired join results with a rank value at least equal to the lowest rank value generated. Experiments evaluate its performance in terms of the number of transmitted tuples and the miss rate. Comparison with the traditional threshold algorithm for centralized systems and other top-K ranking algorithms for P2P networks shows the features of the proposed approach. Copyright © 2006 John Wiley & Sons, Ltd.

Received 25 August 2006; Accepted 27 August 2006

KEY WORDS: join query; Knowledge Grid; peer-to-peer; peer data management; semantic link; top-K query processing

1. INTRODUCTION

With the rapid development of peer to peer (P2P) systems, peer data management systems (PDMSs) have become a promising area [1–3]. A PDMS consists of many autonomous, dynamic and heterogeneous nodes that can exchange data and services in a decentralized and distributed manner. Each site has equal functionality and is both a client and a server. Usually, a P2P system locally controls data and has local knowledge of available data and schemas.

*Correspondence to: Jie Liu, Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, People's Republic of China.
†E-mail: lj@kg.ict.ac.cn

Contract/grant sponsor: National Science Foundation of China; contract/grant number: 60503047
Contract/grant sponsor: National Basic Research Program of China; contract/grant number: 2003CB317000

PDMSs can be classified as unstructured or structured according to the topology of the underlying P2P networks. Unstructured PDMSs use peer clustering and peer indexing approaches rather than flooding or random selection for query routing. Structured PDMSs use a distributed hash table (DHT) for query routing, which can find information within a bounded number of hops in large-scale P2P networks but it does not support complex queries.

PDMSs provide a scalable basis for building Semantic Web applications. However, existing P2P systems do not have the data management capabilities that are typically found in relational databases [4–7]. An important issue is how to efficiently return the top-K results rather than all the satisfactory answers from multiple data sources with the minimum transmission cost. Usually K is quite small compared to the total number of satisfactory answers.

Example 1. Figure 1 is an example of a top-K join query, where data from the ISI Web of Knowledge (ISI) and the Journal Citation Reports (JCR) are located at different peers. A query to retrieve the top 100 most frequently cited papers published by the journals with the highest impact factors can be represented as:

SELECT *Title*, *Author*, *Journal*, *Times Cited*, *Impact Factor*
FROM *ISI*, *JCR*
WHERE *ISI. Journal* = *JCR. Journal*
ORDER BY f(*TimesCited*, *ImpactFactor*)
STOP AFTER 100

The ORDER BY statement invokes a monotonic increasing rank function which takes attributes *TimesCited* and *ImpactFactor* as parameters to calculate the weighted rank value of paper citation and journal impact factor. The STOP AFTER clause asks for 100 ordered join results.

To answer this query, a straightforward way is to join all of the data in ISI with the data in JCR, and then to select the 100 highest ranking papers. However, this approach needs a large bandwidth to make all of the joins. To reduce the amount of data actually involved in the join operation, in this paper we propose a semantic-link-based approach to answer top-K join queries in P2P networks. Pruning is used to remove irrelevant tuples before join matching.

2. RELATED WORK

The idea of top-K queries was introduced early by Fagin [8]. Assume that each object in a database has m attributes, and that for each attribute there is a sorted list of objects. A top-K query returns the objects with the top-K highest overall grade, a combination of the attribute grades by using a monotonic aggregation function. To improve the Fagin algorithm (FA), the threshold algorithm (TA) was proposed independently by at least three groups, including Nepal and Ramakrishna [9], Güntzer *et al.* [10], and Fagin *et al.* [11]. Fagin *et al.* compared the three algorithms in these three papers, and proved that the FA algorithm in [11] is optimal over the algorithms that do not make wild guesses, which perform random access on objects that have not been previously visited by sorted access. Fagin *et al.* [12] defined various distance measures between top-K lists, proposed the equivalence class of

DOI: 10.1002/cpe

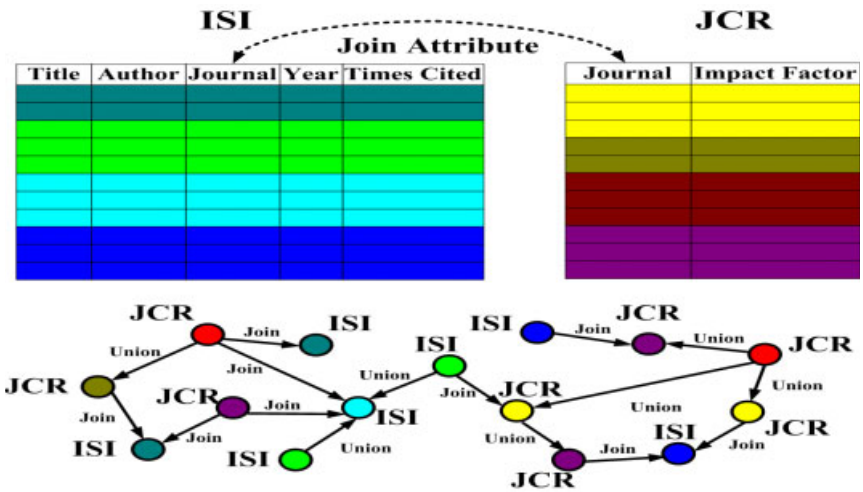


Figure 1. An example of a top-K join query in P2P networks.

distance measures, and used the distance measures to compare the top-K outputs of several popular search engines.

Marian *et al.* [13] studied the processing of top-K queries over autonomous Web-accessible sources. Building on sequential top-K query processing techniques, parallel algorithms satisfying source access constraints were introduced to lower the query response time by increasing source access parallelism. Experiments show that parallel algorithms are significantly more efficient than sequential algorithms.

Mamoulis *et al.* [14] proposed a solution to answering top-K online analytical processing (OLAP) queries by making use of the aggregate R-tree. A branch-and-bound technique is used to compute the query result by following the entries with the highest probability of contributing to large aggregate results.

Traditional offline join algorithms are designed to lessen the time to complete a query. However, ripple join is a new family of algorithms designed to reduce the time before an acceptably precise estimate of the query result is available [15]. Using the basic idea of the ripple join, a rank-join algorithm has been proposed to support top-K join queries in relational databases [16]. The top-K query is optimized by determining the best sequence to perform join operations. Experimental evaluation compares the performance of the proposed approach with that of the J^* algorithm, the first efficient algorithm that can support ordered joins based on user-defined join predicates and handle multiple levels of such joins [17].

A new pipelined query operator NRA-RJ is introduced in [18] to support a nested join hierarchy. The performance study shows that for joining multiple rankings of the same set of objects, NRA-RJ outperforms other pipelined rank join operators, while in the case of an arbitrary join condition, the J^* operator is the best feasible choice. An early hash join algorithm was proposed in [19], which can be dynamically configured by the optimizer with both a rapid response time and a fast overall

DOI: 10.1002/cpe

execution time. Experiments show that the algorithm proposed in [19] outperforms other early hash-join algorithms in overall execution time.

Donjerkovic and Ramakrishnan [20] proposed a probabilistic optimization framework for minimizing the overall cost of top-K queries by considering the optimizer's imprecise knowledge of data distribution and selectivity estimates. Bruno *et al.* [21] used the database histograms to map a multi-attribute top-K selection query to a traditional range selection query that can be optimized and executed by relational database management systems (RDBMSs). The goal is to obtain the K tuples that are the best from relation R for a query q according to a predefined distance function *Dist*. Ilyas and Aref [22] proposed a rank-aware query optimization framework using a probabilistic model for estimating the minimum input size of rank-join operators. The cost of a rank-join operator is determined by (1) the number of required results K and how K is propagated in the pipeline, (2) the number of tuples from inputs containing enough information to produce the top-K answers, and (3) the join selectivity.

PREFER is a prototype for efficiently producing the top-K results of preference queries by using materialized views [23]. The key idea is to find a materialized view whose weight function is similar to the weight function of the query. Angiulli and Pizzuti [24] proposed a novel approximate algorithm to calculate the join query of the top-K closest pairs of two large and high-dimensional data sets according to one of the Minkowski metrics. During a scan, each point from one data set is compared with its closest points in the other data set according to the space filling curve order. RankSQL is a system for supporting efficient evaluations of top-K queries in relational databases [25]. As an extension to the relational algebra, a rank-relational model is proposed to capture the ranking property.

Cao and Wang [26] introduced a three-phase uniform threshold algorithm (TPUT) for answering top-K queries in large-scale networks. The algorithm has three main steps: (1) determine a lower-bound estimate for the Kth value; (2) use the estimate to prune away as many ineligible objects as possible; and (3) inspect the resulting set of objects in all nodes to identify the top-K objects. The performance is improved by the low latency and the low bandwidth consumption.

In the area of P2P networks, Aberer and Wu [27] proposed a ranking algebra as a formal framework for rank computation, which is the first step towards a completely decentralized P2P search engine offering meaningful and efficient rankings. The P2P ranked selection algorithm PSel was proposed by Zhao *et al.* [28] for answering top-K join queries by multicasting the original query together with a lower bound of the rank value. Other P2P top-K ranking approaches include a super-peer-based retrieval algorithm for processing top-K queries [29] and an approximate top-K algorithm KLEE for distributed query processing [30].

This paper extends our previous work [31] by establishing semantic links to represent the semantic relationships between peers' data schemas. A pruning-based approach is proposed for efficiently supporting top-K join queries in P2P networks.

3. GENERAL ARCHITECTURE

Let R and S be two relations scattered through P2P networks, assuming that tuples are distinct and in descending order. A top-K join query can be denoted as Top-K $(R \bowtie_{r(a)\theta s(b)} S)$, where $r(a)$ and $s(b)$ are the join attributes satisfying a join condition $r(a)\theta s(b)$. Let $f(r(p), s(q)) \in [0, 1]$ be a predefined monotonic increasing rank function, where $r(p)$ and $s(q)$ are the rank attributes of R and S. The top-K

DOI: 10.1002/cpe

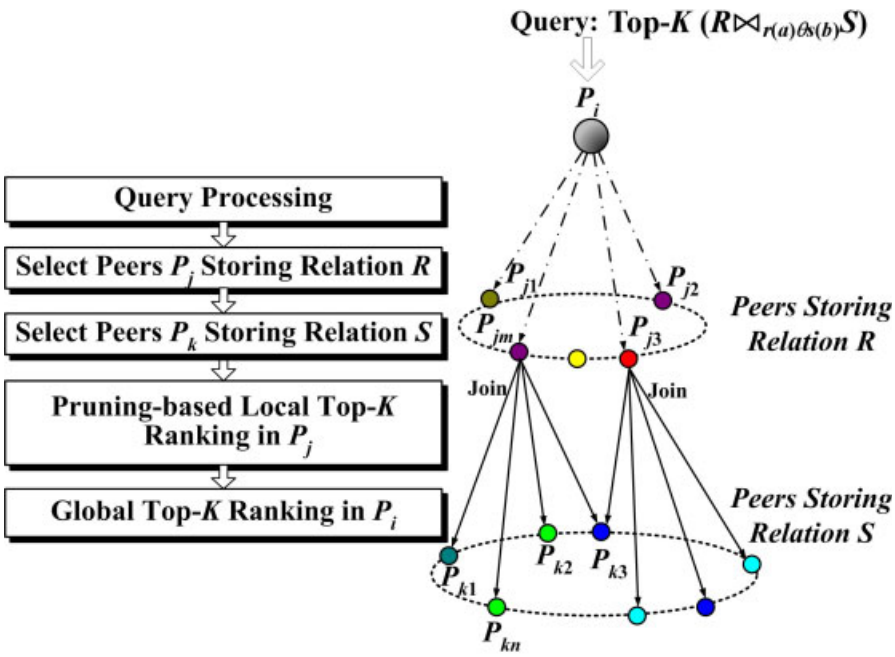


Figure 2. General architecture of semantic-link-based top-K join queries in P2P networks.

join query returns the first K answers ordered by the rank value calculated by the monotonic rank function f.

The general architecture of semantic-link-based top-K join queries is shown in Figure 2. Upon receiving a Top-K $(R \bowtie_{r(a)\theta s(b)} S)$ query, peer P_i performs the following steps to obtain the answers.

(1) *Query processing*. P_i will first parse the query and obtain the rank function, the parameters of the *Select-List*, the join condition $r(a)\theta s(b)$ and the number of ordered results K.
(2) *Select peers storing relation R*. P_i will send a SOAP message Top-K(*Select-List*, $r(a)\theta s(b)$, $f(r(p), s(q))$, T, K) to its neighbors P_j $(j = 1 \ldots m)$. If $r(p) \in$ *Schema* (P_j), $r(a) \in$ *Schema*(P_j), and *Schema*$(P_j) \cap$ *Select-List* $\neq$ NULL, then P_j is selected as one of the join candidates for relation R.
(3) *Select peers storing relation S*. P_i will send a SOAP message Top-K (*Select-List*, $r(a)\theta s(b)$, $f(r(p), s(q))$, T, K) to its neighbors P_k $(k = 1 \ldots n)$. If $s(q) \in$ *Schema*(P_k), $s(b) \in$ *Schema*(P_k), and *Schema*$(P_k) \cap$ *Select-List* $\neq$ NULL, then P_k is selected as one of the join candidates for relation S.
(4) *Pruning-based local top-K ranking*. Peer P_j storing relation R will perform the join operation with peer P_k storing relation S to select the local top-K results by using the pruning-based approach described in Section 5.

DOI: 10.1002/cpe

(5) *Global top-K ranking*. After a predefined time-to-live (TTL), P_j $(j = 1 \ldots m)$ will send the local top-K rank value to P_i, which will then sort all of the returned rank values and send the final lower bound T_{final} to P_j $(j = 1 \ldots m)$. Finally, the join results in P_j $(j = 1 \ldots m)$ with a rank value greater than T_{final} will be sent to P_i to produce the global top-K answers.

4. SEMANTIC LINK MODEL

Let *Schema* (P_i) and *Schema* (P_j) be the data schemas of peer P_i and peer P_j. A semantic link between P_i and P_j is a pointer with the type α directed from the predecessor P_i to its successor P_j, and α can be one of the following.

(1) *Union link*, P_i—*Union* $\rightarrow P_j$, indicates $Schema(P_i) = Schema(P_j)$, i.e. data in P_i and P_j satisfy a union condition in relational model.
(2) *Inclusion link*, P_i—*Inclusion* $\rightarrow P_j$, indicates $Schema(P_i) \supset Schema(P_j)$, i.e. the attribute set of P_i includes P_j's attribute set.
(3) *Extension link*, P_i—*Extension* $\rightarrow P_j$, indicates $Schema(P_i) \subset Schema(P_j)$, i.e. the attribute set of P_i is included in P_j's attribute set.
(4) *Overlap link*, P_i—*Overlap* $\rightarrow P_j$, indicates $Schema(P_i) \cap Schema(P_j) \neq \text{NULL}$, and $Schema(P_i) \not\subset Schema(P_j)$ and $Schema(P_j) \not\subset Schema(P_i)$.
(5) *Join link*, P_i—*Join*$(\delta) \rightarrow P_j$, indicates that $Schema(P_i)$ and $Schema(P_j)$ satisfy a join condition in relational model, and δ is the join selectivity factor [32].
(6) *Empty link*, P_i—$\emptyset \rightarrow P_j$, indicates that there is no semantic relationship between $Schema(P_i)$ and $Schema(P_j)$.

When a peer P_i joins a P2P network, it will randomly take a peer P_j as one of its neighbors and then send a SOAP message to P_j to obtain $Schema(P_j)$. By identifying the semantic relationships between $Schema(P_i)$ and $Schema(P_j)$, the semantic links between P_i and P_j can be established. Semantic links between P_i and other peers can be derived according to the heuristic reasoning rules as proposed in [31]. Before a peer leaves a P2P network, it will ask its predecessors and successors to delete the semantic links between them. To bring the semantics links up to date, each peer issues a 'Stabilization' call periodically in the background to have the semantic link types and predecessor and successor pointers updated. Peers will autonomously inform their predecessors and successors of the new data schemas through messages.

5. PRUNING-BASED LOCAL TOP-*K* RANKING

5.1. Basic idea

The pruning-based top-K ranking approach takes the following parameters as input: relation R, relation S, the join condition $r(a)\theta s(b)$, the monotonic increasing rank function $f(r(p), s(q))$, and the number of desired answers K, where $r(a)$ and $s(b)$ are the join attributes, and $r(p)$ and $s(q)$ are the rank attributes of R and S.

DOI: 10.1002/cpe

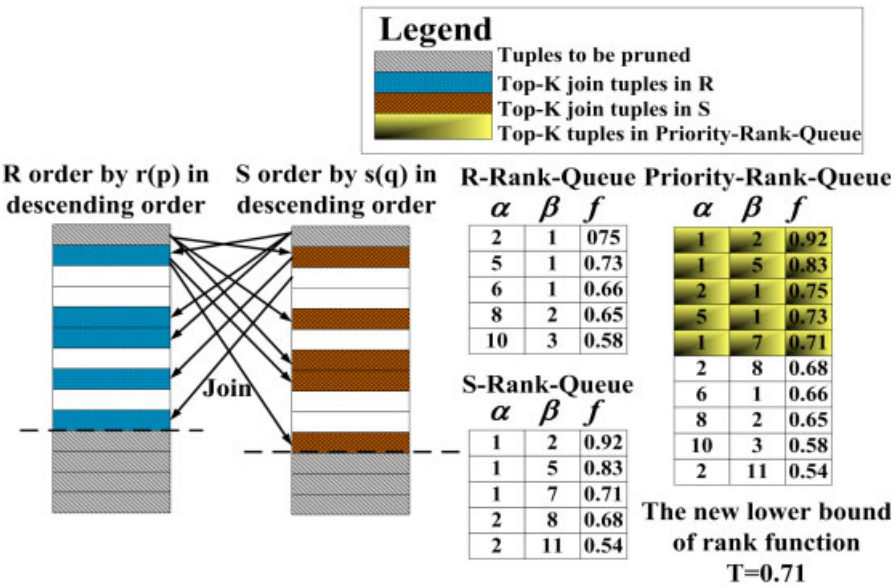


Figure 3. Pruning-based top-K ranking.

Assuming that relation R and relation S are sorted on their rank attributes in descending order, the basic idea of the pruning-based approach is to produce the top-K ranked results by iteratively pruning irrelevant tuples that cannot produce any join results with a rank value greater than or equal to the rank value generated so far. To illustrate the proposed approach, the following items are depicted in Figure 3.

(1) T: the lower bound of the rank function f.
(2) α: the row number of the current tuple in R.
(3) β: the row number of the current tuple in S.
(4) Top-K join tuples in R: the top-K tuples in R that can be joined with the first tuple in S. If the number of tuples in R that can be joined is less than K, then repeat the join process with the next tuple in$S$ until the K join tuples are generated or the end of S is reached.
(5) Top-K join tuples in S: the top-K tuples in S that can be joined with the first tuple in R. If the number of join tuples is less than K, then repeat the join process with the next tuple in R until the K join tuples are generated or the end of R is reached.
(6) *R-Rank-Queue* (α, β, f): the queue holding the row number of top-K join tuples in R, the corresponding row number of joined tuples in S, and the rank value of the join results.
(7) *S-Rank-Queue* (α, β, f): the queue holding the corresponding row number of joined tuples in R, the row number of top-K join tuples in S, and the rank value of the join results.
(8) *Priority-Rank-Queue* (α, β, f): the priority queue holding the union of *R-Rank-Queue* and *S-Rank-Queue*.

DOI: 10.1002/cpe

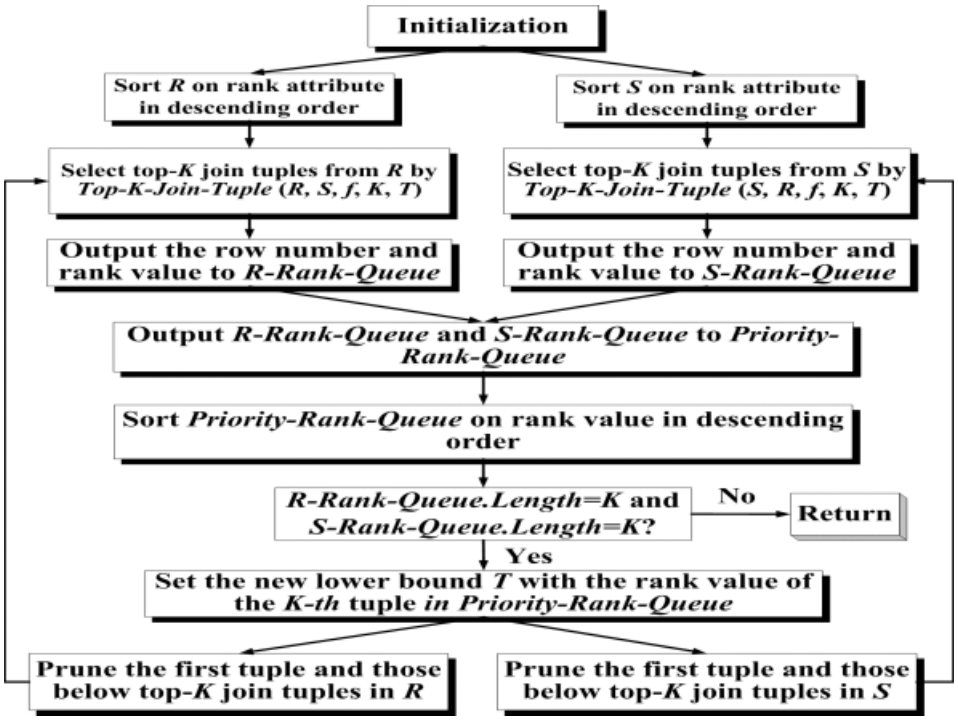


Figure 4. General architecture of the pruning-based local top-K ranking.

5.2. Local top-K ranking approach

The general architecture of the local top-K ranking approach is shown in Figure 4, which includes the following steps to select top-K join results from relation R and relation S.

Step 1. (1) Initialize the lower bound of the rank function as $T = 0$.
(2) Empty *R-Rank-Queue* (α, β, f), *S-Rank-Queue* (α, β, f), and *Priority-Rank-Queue* (α, β, f).

Step 2. Sort R and S on their rank attributes in descending order.

Step 3. Select top-K join tuples from R and S, respectively, by invoking algorithm *Top-K-Join-Tuple* (R, S, f, K, T) and *Top-K-Join-Tuple* (S, R, f, K, T) in Figure 5.

Step 4. (1) Put the row number of the top-K join tuples from R, the row number of the joined tuples from S, and the rank value of the join results into *R-Rank-Queue* (α, β, f).
(2) Put the row number of the joined tuples from R, the top-K join tuples from S, and the rank value of the join results into *S-Rank-Queue* (α, β, f).

Step 5. Put *R-Rank-Queue* (α, β, f) and *S-Rank-Queue* (α, β, f) into *Priority-Rank-Queue* (α, β, f).

DOI: 10.1002/cpe

```
Algorithm Top-K-Join-Tuple (R, S, f, K, T)
Input: the relation R, the relation S, the rank function f, the number of join tuples K, and the
       lower bound T of the rank function.
Output: top-K tuples from R that can be joined with tuples from S.
Begin
 k := 0; //Number of tuples in R that have a join candidate in S
 u := 0; //Row number of the current tuple in S
 While k < K and u < S.Length
    u := u + 1;
    v := 0; // Row number of the current tuple in R
    While k < K and v < R.Length
          v := v + 1;
      If tuple S (u) and tuple R (v) satisfy the join condition and f(R(v).r(p), S(u).s(q)) is
          greater than T
      Then
          Output (v, u, f) to the rank queue of R;
          k := k + 1;
      End If
    End While
 End While
End
```

Figure 5. Algorithm for selecting top-K join tuples.

Step 6. Sort *Priority-Rank-Queue* on the rank value in descending order.

Step 7. If the length of *R-Rank-Queue* or of *S-Rank-Queue* is not equal to K, which implies that no further join results with greater rank value will be produced, then the top-K ordered tuples in *Priority-Rank-Queue* are the top-K rank values for $R \bowtie S$, and the top-K ranking answers will be successfully returned.

Step 8. Set the new lower bound T with the rank value of the Kth tuple in *Priority-Rank-Queue*.

Step 9. (1) Prune the first tuple in R and S, respectively.
(2) Prune the tuples in R below the top-K join tuples because the rank value produced by these tuples joined with tuples in S cannot be greater than or equal to the minimum rank value in *R-Rank-Queue*.
(3) Prune the tuples in S below the top-K join tuples in the same way.

Step 10. Return to Step 3.

The algorithm for selecting top-K join tuples from one relation R that can be joined with tuples from relation S is shown in Figure 5.

Theorem 1. *Using a monotonic combining function, the pruning-based local top-K ranking algorithm correctly returns the top-K join results ordered on the rank function.*

Proof. It can be proved by contradiction. Let *R-Rank-Queue* (α, β, f) be the queue of the top-K join tuples in R, $\alpha_{\min}$, $\beta_{\min}$, and f $(\alpha_{\min}, \beta_{\min})$ be the row number of tuples in R and S with the minimum rank value in the top-K join results. Assume that there exists a join combination of the ith tuple in R and the jth tuple in S satisfying $\alpha_{\min} < i$, $\beta_{\min} < j$, and $f(\alpha_{\min}, \beta_{\min}) < f(i, j)$. Because each relation

DOI: 10.1002/cpe

R and S is ranked in descending order, $\alpha_{\min} < i$, and the rank function is monotonic increasing, then $f(i, \beta_{\min}) < f(\alpha_{\min}, \beta_{\min}) < f(i, j)$, we have $j < \beta_{\min}$, which contradicts the original assumption $\beta_{\min} < j$. So, the join combination (i, j) does not exists, and tuples below $\alpha_{\min}$ in R can be pruned.

In the same way, we conclude that the *S-Rank-Queue* holds the top-K join tuples in S, and tuples pruned in R and S cannot produce a join matching with a rank value greater than the rank value generated, so the top-K join results are among the remaining tuples. □

Theorem 2. *The pruning-based local top-K ranking algorithm requires only bounded buffers, whose size is independent of the data stored at each peer.*

Proof. All that the pruning-based local top-K ranking algorithm must keep in a buffer is the current top-K join tuples, their grades, and the priority queue. Other than the space required to perform the join operation, the required space is independent of the size of the input at each peer. □

6. PERFORMANCE ANALYSIS

To illustrate and evaluate the performance of the proposed approach, we have simulated an unstructured P2P semantic link network, which consists of 100 peers. Two synthetic data sets R (*RID*, *FID*, $K1$) and S (*SID*, $K2$) are distributed uniformly over each peer. Each relation includes 10 000 tuples, where *RID* and *SID* are the key, *FID* is the foreign key in relation R corresponding to *SID* in relation S, and $K1$, $K2 \in [0, 1]$ are the rank values of some rank attributes in relations R and S. The rank function is $f = \alpha \times K1 + \beta \times K2$. Herein, we assume that $K1$ and $K2$ play the same role in the rank function and take the weights $\alpha = 0.5$ and $\beta = 0.5$ in the simulation. The evaluation metrics are (1) the number of transmitted tuples and (2) the miss rate for answering a top-K join query. To avoid network flooding, each peer selects a number of peers as its neighbors to forward a query. The neighbors with higher join selectivity factor have priority to be selected as a join candidate.

The first experiment measures the number of transmitted tuples. After receiving a top-K join query request, peer P_i will perform the following steps to answer the query.

Step 1. Parse the query to obtain the parameters of the *Select-List*, the *join condition* $r\ (a)\theta s(b)$, the rank function f and the number of ordered results K.
Step 2. Forward the top-K query request to the neighbors through a *Join* or *Union* semantic link.
Step 3. Join with its neighbors to produce local top-K join results by using the proposed pruning-based approach.
Step 4. Return the local top-K rank value to the query initiator.
Step 5. After a predefined time, the query initiator will sort all of the rank values returned and set the final lower bound T_{final}.
Step 6. After receiving T_{final}, tuples in the join candidates with a rank value greater than T_{final} will be returned to the query initiator as the global top-K join results.

Figure 6 compares the number of transmitted tuples produced by the proposed approach when different neighbors are selected through *Join* semantic links. For each K, the number of transmitted tuples is an average of ten rounds. As shown in Figure 6, when each peer selects five neighbors to forward a top-K query, the average number of transmitted tuples is 39 967. When the total number of neighbors varies from 10, 15, to 20, the average number of transmitted tuples is 81 352, 142 897, and

DOI: 10.1002/cpe

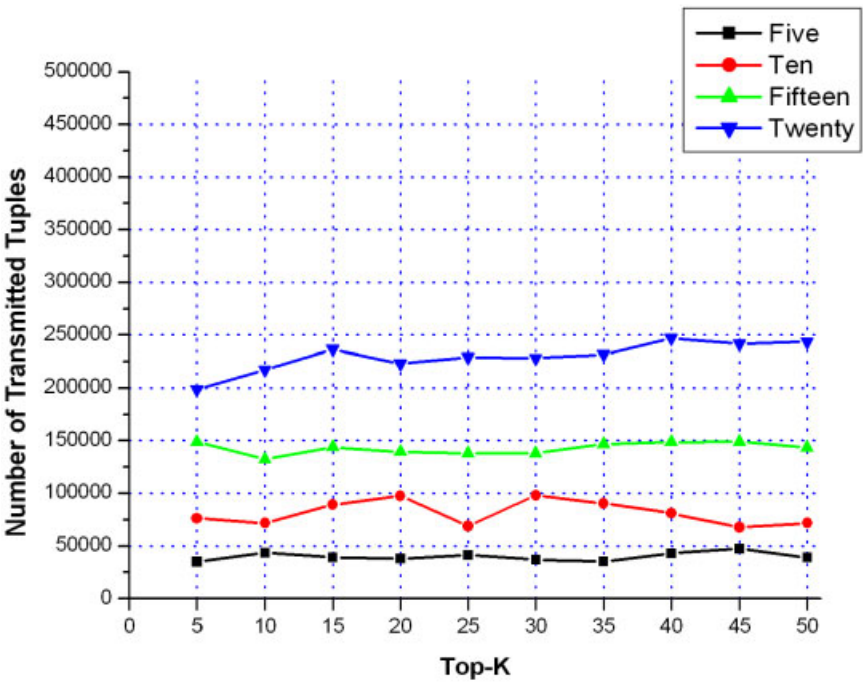


Figure 6. Number of tuples transmitted for answering top-K join queries.

229 722, respectively. This experiment shows that the greater the number of neighbors selected, the greater the number of tuples transmitted.

The second experiment counts the number of top-K join results not retrieved by the proposed approach but which can be found by joining relation R and relation S globally. The top-K miss rate is defined as

$$\frac{\text{number of top join results missed}}{K}$$

where the number of top join results missed is the total number of top-K results that are not retrieved by the proposed approach, and K is the required total number of join results with the highest rank value. Figure 7 plots the miss rate for answering a top-K join query. For each K, the miss rate is an average of 10 rounds. When the number of neighbors varies from 20, 15, 10, to 5, the average miss rate varies from 28.4%, 26.2%, 23.2%, to 21.3%. This experiment shows that greater the number of neighbors selected, the lower the top-K miss rate.

Figure 8 depicts the relationship between the number of transmitted tuples and the miss rate. The x-axis is the average miss rate when the selected neighbors are 5, 10, 15, and 20, and the y-axis denotes the average number of transmitted tuples. This experiment shows that the miss rate

DOI: 10.1002/cpe

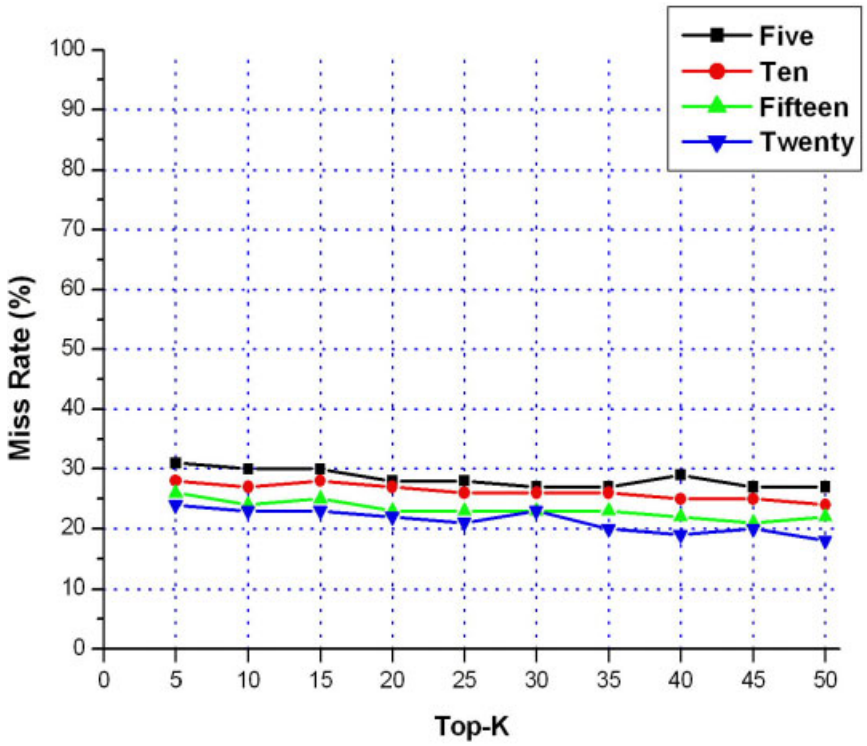


Figure 7. Miss rate for answering top-K join queries.

monotonically decreases with the number of transmitted tuples. Asking for a lower miss rate will result in heavy traffic and high transmission cost in P2P networks.

7. DISCUSSION

In [22], two major criteria are proposed to classify the rank aggregation algorithms. The first classification is based on how the ranked inputs are used, by sorted access or by random access. Sorted access uses objects in the order of their scores. Random access probes or queries an input to retrieve the score of a given object. The second classification is based on whether all of the inputs contain the same set of objects or not. The proposed semantic-link-based top-K join approach allows both sorted access and random access, but the data objects being accessed vary in content at each peer.

As summarized in Table I, the differences between the proposed top-K join approach and the TA proposed by Fagin *et al.* [11] are as follows.

DOI: 10.1002/cpe

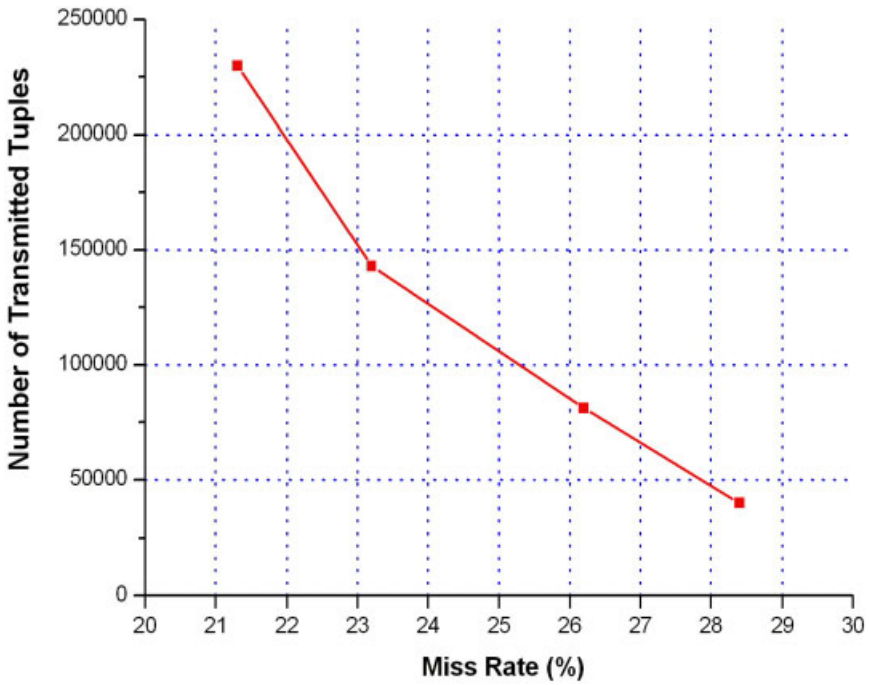


Figure 8. The relationship between the number of transmitted tuples and the miss rate.

Table I. Comparison between the proposed approach and the TA.

	Semantic-link-based approach	TA
Scenario	P2P networks	Centralized
Query type	Top-K join	Top-K
Number of sorted lists	Unknown and relatively large	Known and relatively small
Number of objects in each sorted list	Unknown	Known
Data objects	Different on each peer	Same set
Query results	Approximate top-K	Accurate top-K

(1) The proposed approach is a solution for answering top-K queries in distributed P2P networks, while the TA is suitable for centralized systems.
(2) The proposed approach supports top-K join queries involving multiple data sources, while the TA only supports top-K queries without any join operations.
(3) The number of sorted lists is unknown and relatively large in P2P networks, while it is known and relatively small for the TA.

DOI: 10.1002/cpe

(4) The number of objects in each sorted list is unknown in P2P networks, while it is known for the TA.
(5) The data objects accessed by the proposed approach vary in content at each peer, while they are the same for the TA.
(6) Because of the autonomous and dynamic characteristics of P2P networks and the requirement of TTL, the query results returned by the proposed approach are approximate top-K results; however, the top-K results returned by the TA are accurate.

In the following we compare the proposed semantic-link-based top-K join approach with other top-K ranking approaches in P2P networks.

An algorithm PSel for answering ranked join queries in DHT-based P2P networks was proposed in [28], with two main steps: (1) calculating a lower bound on the rank value using the sampling method; (2) pruning irrelevant tuples with a rank value below the lower bound before join matching. However, the performance of the PSel algorithm depends greatly on the lower bound set by the sampling approach.

A distributed top-K retrieval algorithm for super-peer-based P2P networks was proposed in [29], where super-peers are responsible for top-K query processing. However, the proposed approach was based on the *HyperCube* topology, and it is a solution for top-K queries but not for top-K join queries.

Hose *et al*. proposed an efficient approach for processing top-K queries on structured data in P2P-based systems [33]. For answering a top-K query, the algorithm misses at most M data items of the globally correct top-K results with a probabilistic guarantee. There are two limitations to this approach. First, the authors assume that each participating peer has gathered global knowledge in the histogram of the query attributes, which results in high maintenance costs in P2P networks. Secondly, only a one-dimensional histogram is considered, and this is not suitable for answering top-K join queries on multiple attributes.

Compared to previous work on top-K ranking approaches in P2P networks, the distinctive features of the approach proposed here are as follows.

(1) Semantic links are used to denote the relationships between peers' data schemas; this has a relatively lower maintenance cost than the approaches indexing on peers' data content. Moreover, the semantic links between any two peers can be derived according to a set of reasoning rules if there are some relationships between them.
(2) The proposed approach provides a good solution for answering top-K join queries in unstructured P2P networks, which requires no global information and makes no assumption about the topology of the underlying P2P networks.
(3) The number of transmitted tuples can be efficiently reduced by pruning irrelevant tuples before join matching.

8. CONCLUSION

In this paper we have proposed a semantic-link-based infrastructure for efficiently processing top-K join queries in P2P networks. A pruning-based local top-K ranking approach has been proposed to reduce the total amount of transmitted data by removing irrelevant tuples as early as possible in query processing. The performance of the proposed approach has been given by the number of transmitted

DOI: 10.1002/cpe

tuples and the top-K miss rate. The proposed approach can be used in the P2P-based Knowledge Grid [34–36] to support advanced applications in P2P knowledge management.

Our work will continue in three main areas. First, we plan to incorporate query optimization techniques, such as bloom filters and ripple join algorithms, with the proposed top-K join approach to reduce the transmission cost, the response time, and the miss rate. Secondly, we plan to apply the proposed approach to answering top-K join queries in XML. Finally, we plan to look for a good method of answering top-K join queries in structured P2P networks.

ACKNOWLEDGEMENTS

This research work was supported by the National Science Foundation of China (No. 60503047) and the National Basic Research Program of China (No. 2003CB317000).

REFERENCES

1. Aberer K, Cudre-Mauroux P. Semantic overlay networks. *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*, Trondheim, Norway, 30 August–2 September 2005. ACM Press: New York, 2005; 1367.
2. Bernstein P, Giunchiglia F, Kementsietsidis A, Mylopoulos J, Serafini L, Zaihrayeu I. Data management for peer-to-peer computing: A vision. *Proceedings of the 5th International Workshop on the Web Databases (WebDB 2002)*, Madison, WI, 6–7 June 2002; 89–94.
3. Ooi B, Tan K. Guest editors' introduction: Special section on peer-to-peer-based data management. *IEEE Transactions on Knowledge and Data Engineering* 2004; **16**(7):785–786.
4. Hellerstein J. Architectures and algorithms for Internet-scale (P2P) data management. *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB 2004)*, Toronto, Canada, 31 August–3 September 2004. Morgan Kaufmann: San Francisco, CA, 2004; 1244.
5. Kementsietsidis A, Arenas M, Miller R. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. *Proceedings of the ACM SIGMOD 2003 International Conference on Management of Data*, San Diego, CA, 9–12 June 2003. ACM Press: New York, 2003; 325–336.
6. Koloniari G, Pitoura E. Peer-to-peer management of XML data: Issues and research challenges. *ACM SIGMOD Record* 2005; **34**(2):6–17.
7. Tatarinov I, Halevy A. Efficient query reformulation in peer data management systems. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Paris, France, 13–18 June 2004. ACM Press: New York, 2004; 539–550.
8. Fagin R. Combining fuzzy information from multiple systems. *Proceedings of the 15th ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems (PODS 1996)*, Montreal, Canada, 3–5 June 1996. ACM Press: New York; 1996; 216–226.
9. Nepal S, Ramakrishna MV. Query processing issues in image (multimedia) databases. *Proceedings of the 15th International Conference on Data Engineering (ICDE 1999)*, Sydney, Australia, 23–26 March 1999. IEEE Computer Society Press: Los Alamitos, CA, 1999; 22–29.
10. Güntzer U, Balke W, Kießling W. Optimizing multi-feature queries in image databases. *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB 2000)*, Cairo, Egypt, 10–14 September 2000. Morgan Kaufmann: San Francisco, CA, 2000; 419–428.
11. Fagin R, Lotem A, Naor M. Optimal aggregation algorithms for middleware. *Proceedings of the 20th ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems (PODS 2001)*, Santa Barbara, CA, 21–23 May 2001. ACM Press: New York, 2001; 102–113.
12. Fagin R, Kumar R, Shivakumar D. Comparing top-k lists. *SIAM Journal of Discrete Mathematics* 2003; **17**(1):134–160.
13. Marian A, Bruno N, Gravano L. Evaluating top-K queries over Web-accessible databases. *ACM Transactions on Database Systems* 2004; **29**(2):319–362.
14. Mamoulis N, Bakiras S, Kalnis P. Evaluation of top-k OLAP queries using aggregate R-trees. *Proceedings of the International Symposium on Spatial and Temporal Databases (SSTD 2005)*, Angra dos Reis, Brazil, 22–24 August 2005. Springer: Berlin, 2005; 236–253.
15. Haas P, Hellerstein J. Ripple joins for online aggregation. *Proceedings of the ACM SIGMOD 1999 International Conference on Management of Data*, Philadelphia, PA, 1–3 June 1999. ACM Press: New York, 1999; 287–298.

DOI: 10.1002/cpe

16. Ilyas I, Aref W, Elmagarmid A. Supporting top-K join queries in relational databases. *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003)*, Berlin, Germany, 9–12 September 2003. Morgan Kaufmann: San Francisco, CA, 2003; 754–765.
17. Natsev A, Chang Y, Smith J, Li C, Vitter J. Supporting incremental join queries on ranked inputs. *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001)*, Roma, Italy, 11–14 September 2001. Morgan Kaufmann: San Francisco, CA, 2001; 281–290.
18. Ilyas I, Aref W, Elmagarmid A. Joining ranked inputs in practice. *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002)*, Hong Kong, China, 20–23 August 2002. Morgan Kaufmann: San Francisco, CA, 2002; 950–961.
19. Lawrence R. Early hash join: A configurable algorithm for the efficient and early production of join results. *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*, Trondheim, Norway, 30 August–2 September 2005. ACM Press: New York, 2005; 841–852.
20. Donjerkovic D, Ramakrishnan R. Probabilistic optimization of top N queries. *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB 1999)*, Edinburgh, Scotland, U.K., 7–10 September 1999. Morgan Kaufmann: San Francisco, CA, 1999; 411–422.
21. Bruno N, Chaudhuri S, Gravano L. Top-K selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Transactions on Database Systems* 2002; **27**(2):153–187.
22. Ilyas I, Aref W. Rank-aware query processing and optimization. *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, Tokyo, Japan, 5–8 April 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 1144.
23. Hristidis V, Koudas N, Papakonstantinou Y. PREFER: A system for the efficient execution of multi-parametric ranked queries. *Proceedings of the ACM SIGMOD 2001 International Conference on Management of Data*, Santa Barbara, CA, 22–24 May 2001. ACM Press: New York; 2001; 259–270.
24. Angiulli F, Pizzuti C. An approximate algorithm for top-K closest pairs join query in large high dimensional data. *Data and Knowledge Engineering* 2005; **53**(3):263–281.
25. Li C, Chang K, Ilyas I, Song S. RankSQL: Query algebra and optimization for relational top-K queries. *Proceedings of the ACM SIGMOD 2005 International Conference on Management of Data*, Baltimore, MD, 14–16 June 2005. ACM Press: New York, 2005; 131–142.
26. Cao P, Wang Z. Efficient top-K query calculation in distributed networks. *Proceedings of the 23rd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2004)*, Newfoundland, Canada, 25–28 July 2004. ACM Press: New York, 2004; 206–215.
27. Aberer K, Wu J. A framework for decentralized ranking in Web information retrieval. *Proceedings of the 5th Asia-Pacific Web Conference (APWeb 2003)*, Xi'an, China, September 2003. Springer: Berlin, 2003; 213–226.
28. Zhao K, Zhou S, Tan K, Zhou A. Supporting ranked join in peer-to-peer networks. *Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA 2005)*, Copenhagen, Denmark, 22–26 August 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 796–800.
29. Balke W, Nejdl W, Siberski W, Thaden U. Progressive distributed top k retrieval in peer-to-peer networks. *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, Tokyo, Japan, 5–8 April 2005. IEEE Computer Society Press: Los Alamitos, CA, 2005; 174–185.
30. Michel S, Triantafillou P, Weikum G. KLEE: A framework for distributed Top-K query algorithms. *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*, Trondheim, Norway, 30 August–2 September 2005. ACM Press: New York, 2005; 637–648.
31. Zhuge H, Liu J, Feng L, Sun X, He C. Query routing in a peer-to-peer semantic link network. *Computational Intelligence* 2005; **21**(2):197–216.
32. Mishra P, Eich M. Join processing in relational databases. *ACM Computing Surveys* 1992; **24**(1):63–113.
33. Hose K, Karnstedt M, Sattler K, Zinn D. Processing top-N queries in P2P-based Web integration systems with probabilistic guarantees. *Proceedings of the 8th International Workshop on the Web and Databases (WebDB 2005)*, Baltimore, MD, 16–17 June 2005; 109–114.
34. Zhuge H. *The Knowledge Grid*. World Scientific: Singapore, 2004.
35. Zhuge H, Sun X, Liu J, Yao E, Chen X. A scalable P2P platform for the Knowledge Grid. *IEEE Transactions on Knowledge and Data Engineering* 2005; **17**(12):1721–1736.
36. Zhuge H. A Knowledge Grid model and platform for global knowledge sharing. *Expert Systems with Applications* 2002; **22**(4):313–320.

DOI: 10.1002/cpe