Ranking in Distributed Uncertain Database Environments

Yousry M. Abdul Azeem, Ali I. ElDesouky, and Hesham A. Ali
Computers Engineering and Systems Department, Faculty of Engineering, Mansoura University, Egypt

Article Info

Article history:

Received Mar 4, 2014
Revised Jun 26, 2014
Accepted Jul 10, 2014

Keyword:

Database applications
Distributed databases
Ranking
Top-*k* query

ABSTRACT

Distributed data processing is a major field in nowadays applications. Many applications collect and process data from distributed nodes to gain overall results. Large amount of data transfer and network delay made data processing in a centralized manner a hard operation representing an important problem. A very common way to solve this problem is ranking queries. Ranking or top-*k* queries concentrate only on the highest ranked tuples according to user's interest. Another issue in most nowadays applications is data uncertainty. Many techniques were introduced for modeling, managing, and processing uncertain databases. Although these techniques were efficient, they didn't deal with distributed data uncertainty. This paper deals with both data uncertainty and distribution based on ranking queries. A novel framework is proposed for ranking distributed uncertain data. The framework has a suite of novel algorithms for ranking data and monitoring updates. These algorithms help in reducing the communication rounds used and amount of data transmitted while achieving efficient and effective ranking. Experimental results show that the proposed framework has a great impact in reducing communication cost compared to other techniques.

Corresponding Author:

Yousry M. Abdul Azeem,
Computers Engineering and Systems Department, Faculty of Engineering, Mansoura University
Daqahlia 35516, Egypt
Email: yousry@mans.edu.eg

1. INTRODUCTION

Distributed data processing is a major field in nowadays applications. Most of the common applications collect and process data from distributed sites to gain an overall result. Examples of such applications are; Content Distribution Network (CDN) [1, 2], sensor networks [3, 4, 5, 6], multimedia database [7], information retrieval from geographically separated data centers [8, 9, 10], network monitoring over distributed logs [11] and data extracted from a set of data streams [12, 13]. Centralized processing of sites' data is a very expensive task. Moreover, large amount of data transfer and network delay makes it hard to manipulate these applications in a centralized manner [1]. Usually, in most of these applications, the user does not need to process all data in the system. Instead, only a fraction of data is returned to the user according to his interest. This is often done by using ranking queries. Ranking or top-*k* queries return only the highest ranked *k* tuples according to a user-defined scoring function. Many techniques were designed to rank distributed data introducing many efficient algorithms [3, 7, 8, 9, 11, 12]. However, none of these techniques has dealt with distributed uncertain data. Even though, in most of the nowadays applications, data is fuzzy or uncertain [14, 15, 16, 17, 18, 19, 20, 21].

Top-*k* queries in uncertain database system aim at finding the highest ranked *k* tuples over all possible in- stances of the database. These instances are called possible worlds. Processing this query will consume both time and communication bandwidth. Many recent approaches have dealt with ranking centralized uncertain database [22, 23, 24, 25]. However, there are only two approaches that have dealt with ranking in distributed uncertain systems [15, 26]. These were the first approaches to address ranking in uncertain distributed environment, but they had some drawbacks. In the first approach the main concern was the transmission bandwidth trading-off latency. The central server communicates with other nodes several times per one tuple each access, for computing top-*k* list. This large number of accesses increases the latency. The second approach was applied mainly on wireless sensor networks that are divided into clusters. Each cluster head sends pre-pruned set of tuples (sufficient set) to the query node. Each sufficient set needs a lot of computation which is done at each cluster head. Moreover, both approaches don't monitor updates in tuple values or ranks. This paper proposes a new framework that will help in overcoming these drawbacks. The framework utilizes only one or two rounds of communication in the proposed algorithms. So, the amount of transmitted data is minimized for achieving efficient distributed uncertain database ranking.

The rest of this paper is organized as follows. Section 2 presents the proposed framework and defines the used data model. Section 3 discusses the details of the proposed algorithms. Section 4 validates the proposed algorithms and evaluates their efficiency and performance by a comprehensive experimental study. Finally, the paper is concluded in Section 7.

2. THE PROPOSED FRAMEWORK

The crucial problems facing the current approaches of distributed uncertain database ranking are considered as follows: (*i*) The number of tuples each site chooses to send as a local answer to the query. Large number of tuples consumes both network bandwidth and time; also some tuples may not be involved in the ranking process. On the other hand, small number of tuples may cause inconsistent ranking results. (*ii*) The number of communication rounds (phases) applied in ranking process will affect also the consumed network bandwidth. So, it is required to minimize the number of communication phases. (*iii*) Moreover, The need for monitoring process is also an important issue. Only effective updates in values and probabilities of tuples at different sites should be reported to query node to update the current query answer.

The proposed framework aims to solve the previously discussed problems. The main features of this frame- work are: (*i*) Fixing the number of communication rounds (only one round). (*ii*) Minimizing the number of candidate tuples. Candidate tuples are the set of tuples each node will send to query node for ranking process. (*iii*) Monitoring tuple updates is a main phase after computing the top-*k* list. The framework consists of three main layers, *Query*, *Ranking*, and *M onitoring* layers. In these layers a suite of novel algorithms are introduced and applied.

The *Query* layer is a starting point for the framework in which the distributed top-*k* query is formulated then broadcasted to all nodes. This query may be accompanied with a threshold value. Process execution in this layer takes place at the side of the *Query* node.

In the *Ranking* layer each site, on receiving the query, starts computing the needed tuples for query answer. If the query has a threshold value then a pruning phase is conducted first, otherwise continue with the ranking process. The main phase in this layer is the answer computation phase which is managed entirely using the newly proposed algorithm Threshold-tuned Distributed Top-*k* (TDT*k*). After computing the needed tuples for query answer, each site sends its local answer to the *Query* node to compute the distributed top-*k* list (DT*k*). The *kth* tuple in DT*k* is broadcasted to all the nodes to start the monitoring phase. Process execution in this layer is done at both sides of the *Query* node and other nodes simultaneously.

In the *Monitoring* layer, each node assigns its lower bound tuple with the received one. This tuple is the corner stone in the monitoring process. The tuples are ranked at each site considering lower bound tuple in the ranking process. Tuples ranked lower than lower bound tuple are assigned as candidate list. Any change in candidate list values or any new values updated with a rank lower than lower bound tuple is considered a candidate to be in top-*k*. This tuple is sent to query node to update the current DT*k*. *Monitoring* phase is a continuous process executed on nodes with tuples' updates and *Query* node.

2.1. Data Model Definition

Consider an uncertain database *D*, horizontally distributed over a set of nodes *M* with size |M| = *m* such that $D = \bigcup_{i=1}^{m} D_i$. A central server *R* works as the *Query* node. The database consists of N tuples fragmented with different sizes such that for any site M_i, $|D_i| = N_i$ and $N = \sum_{i=1}^{m} N_i$. Tuples in D_i are denoted as $T_i = \bigcup_{j=1}^{N_i} t_{ij}$ and their score values as random variables$X_i = \bigcup_{j=1}^{N_i} X_{ij}$a random variable X_{ij}'s *pdf* is represented with the pairs (v_{jx}, p_{jx}) where $1 \le x \le B_i$ and B_i is the bounded size of X_{ij}. The main objective of this work is to report at *R* the answer of a Distributed top-*k* query. Formally, Definition 1 Distributed top-k query (*DT*k): It is the query that returns the top-k list of tuples with the highest rank among all distributed nodes. Let D be an uncertain database distributed over M nodes such that $D = \bigcup_{i=1}^{m} D_i$, let *f* be the ranking function. *DT*k query returns the min k tuples such that:

$$DTk = \min_{i=1\ldots k}\left(\bigcup_{x=1}^{M}\{t \in D_x \mid \forall i < j, t_i \prec_f t_j\}\right) \tag{1}$$

Where $ti \prec f\ tj$ means that ti dominates tj according to ranking function f. If the ranking function used is Expected rank then the DT*k*-query returns the top-*k* list with the lowest expected rank among all distributed nodes *M*.

3. THE PROPOSED ALGORITHMS

In this section, we discuss the proposed suite of algorithms which are implemented in our framework. The two proposed algorithms are employed in the *Ranking* layer. Each one of the ranking algorithms is implemented to rank distributed database at attribute-level uncertainty. The first algorithm was introduced in a prior work [27].

3.1. Threshold-tuned Distributed Top-*k* algorithm (TDT*k*)

In this algorithm, a user-defined threshold value is used to prune tuples at each node. The rest tuples are ordered according to their expected rank then the top-*k* list, from each node, is sent to the query node. The query node ranks the tuples to get DT*k* then broadcast the *kth* tuple in DT*k* to all nodes. This value is used to modify local lower bounds used for monitoring phase. Formally, a DT*k* query in TDT*k* algorithm *Qk* returns the highest ranked *k* tuples such that:

$$DTk = \bigcup_{i=1}^{k}\{t_i \in A \mid r(t_i) < (t_{i+1})\} \tag{2}$$

Where:$A = \bigcup_{x=1}^{m} A_x$, A_x is the set of highest score *k*-tuples from site x or : $A_x = \bigcup_{j=1}^{k}\{t_j \in D_x \mid r(t_j) < (t_{j+1})\}$ and $r(t_i)$ as in equation 2 with $p_{ij} \geq \tau$ (the threshold value). The detailed steps of TDT*k* algorithm are as follows: The *Query* node *R* broadcasts a top-*k* query Q_k with pruning threshold value τ, to all nodes. It initializes the priority queue DT*k* and tuple *LT* with empty values. Each node $M_i \in M$ empties a local priority queue A_i and initializes lower bound LBT_i with empty value. Tuples' values with probabilities less than τ are omitted. The tuples are then ranked according to their expected ranks and inserted into A_i. The first *k* tuples in A_i are sent to *R* and inserted in DT*k*. LBT_i is set with the k^{th} tuple in A_i. After computing DT*k*, *R* broadcasts the last tuple *LT* to all other nodes. Each node, sets LBT_i with the received *LT* and starts monitoring phase.

3.2. Ceiling-tuned Distributed Top-k Algorithm (CDTk)

In this algorithm, each node maintains a priority queue with local top-*k* list ranked by expected rank. The first $\left\lceil \frac{k}{m} \right\rceil$ tuples from each node is sent to query node. The rest tuples are called candidate list used later to refine DT*k*. The query node ranks the tuples to get first phase DT*k*. The k^{th} ranked tuple in DT*k* is sent to all nodes of the first (*k* - 1)-tuples in DT*k*. Each node ranks the tuples in candidate list considering the received tuple from *Query* node, then sends all tuples ranked lower than that tuple. Afterwards, Query node ranks the tuple for the second time to get the actual DT*k*. Again, the last tuple in DT*k* is used, but to specify local lower bound in order to start monitoring phase.

A DT*k* query in CDT*k* algorithm is formulated based on equation 3. It returns the highest ranked k tuples with the lowest expected rank among all sites, in two successive phases. Formally,

$$DTk = \bigcup_{i=1}^{k}\{t_i \in A, r(t_i) < (t_{i+1})\} \tag{3}$$

where *A* is the set of tuples with the lowest expected rank from all nodes, $A = \bigcup_{x=1}^{m} A_x$, A_x is the set of tuples with lowest expected rank from node x such that: $A_x = \bigcup_{n=1}^{C}\{t_n \in D_x, r(t_n) < r(t_{n+1})\}$ in the 1st phase and Ax = $\bigcup_{r(t_j)<r(LT)}\{t_j \in D_x\}$ in the 2nd phase, where $C = \left\lceil \frac{k}{m} \right\rceil$, r($t_i$) is expected rank of t_i as in equation 4 and *LT* is the k^{th} ranked tuple in the first phase.

The detailed steps of CDTk algorithm are as follows: The Query node R computes the number of tuples needed from each site C. Afterwards it broadcasts a top-k query (specifying C) to all sites. It initializes the priority queue DTk and tuple LT with empty values. Each node Mi 2 M initializes the priority queue A_i and tuple LBT_i with empty values. It ranks its own tuples according to their expected rank and sends the first *C* tuples to *R* while keeping the rest tuples in A_i (candidate list) for further DT*k* refining. At *R*, tuples are ranked by expected rank then the k^{th} ranked tuple is sent to all the sites of the first (*k*-1)-tuples in DT*k*. Each node assigns LBT_i with *LT* then ranks A_i with LBT_i. Tuples ranked lower than LBT_i are sent to *R*. Another ranking phase is done to get the refined DT_k. The new value of *LT* is computed and broadcasted to all the nodes. Each node sets LBT_i with *LT* value and starts monitoring phase.

4. EXPERIMENTAL STUDY

A data generator is developed in order to generate synthetic datasets used in algorithms verifying. It generates synthetic Gaussian dataset where each record's score attribute draws its values from a Gaussian distribution. For each record, the standard deviation σ is randomly selected from [1,1000] and the mean μ is randomly selected from [5σ,100000]. Each record has *g* choices for its score values where g is randomly selected from 1 to 5. The generator controls both score values and probabilities of the generated tuples.

4.1. Effect of Different Sizes Top-*k* Lists *k*

Figure 1 shows the communication costs of the both proposed algorithms for different values of *m* (10, 50, 100, 200, 500 and 1000 nodes). Each figure compares between the both algorithms representing TDT*k* with zero threshold value (τ = 0). It is observed from figure 1 that the communication cost of TDT*k* increases linearly with *k* until *k* = *N/m* (number of tuples at each site). After that value the communication cost is fixed for any greater *k*. Each site in this case will send all its tuples so the communication cost will be the same. The communication cost of CDT*k* also increases linearly with k but with a lower rate so that it appears, compared to TDT*k*, as if it has a fixed value. Figure 1 also shows that, the communication cost of *A-BF* is fixed for any value of *k*.

4.2. Effect of Different Number of Tuples N

The effect of different *N* (total number of tuples in the database) on the communication cost is studied here for a fixed number of sites on which tuples are distributed. Figure 2a shows the communication cost of the two proposed algorithms with *m* = 10 and *k* = 10; *k* = 100 and different values of *N*. The observed communication cost of TDT*k* increases linearly with *N* until *N* = 1000 then it gets stable (same value with more *N*), while the communication cost of CDT*k* increases linearly with *N*.

The reason of this behavior in TDT*k* is that each site sends k tuples to be processed. At lower values of *N* (when number of tuples at each site is less than *k*) each site sends all its tuples, so that the communication cost increases until *N* = *m* x *k*. Increasing *N* more than *m* x *k* will not affect number of tuples sent which is *k*. On the other hand CDT*k* shows a linear increase with the increase of N although the number of tuples sent is always the same ($\lceil k/m \rceil$). Figure 2b shows the communication cost of the two proposed algorithms but with *m* = 100 and *k* = 10; *k* = 100. TDT*k* has the same behavior as with *m* = 10, while CDT*k* shows a very low communication cost compared to TDT*k*. The distribution of the same number of tuples among larger number of sites increases the probability of getting top-k quickly or at least having a higher lower bound in the beginning of the second phase of the algorithm. Figure 2a and 2b show also a comparison between TDT*k*, CDT*k* and *A-BF* to observe the effect of different *N* on each algorithm. As seen from both figures the communication cost of *A-BF* increases linearly with a very high rate.

4.3. Effect of Different Number of Distributed Sites *m*

The effect of *m* on the communication cost is studied here for a fixed number of tuples distributed on different sites. Figure 2c shows the communication cost of the two proposed algorithms with *N* = 100000 and *k* = 10; *k* = 100 and different values of *m*. The observed communication cost of TDT*k* increases linearly with the increase of *m*, while the communication cost of CDT*k* decreases also linearly with the increase of *m*. The matching point of the two algorithms is at *m* = 150 or *m* = 75. At these values, the communication costs of both algorithms are nearly equal. After these values, TDT*k* tends to increase while CDT*k* decreases. Figure 2c shows that, the communication cost of *A-BF* increases linearly but with higher values than both TDT*k* and CDT*k*.

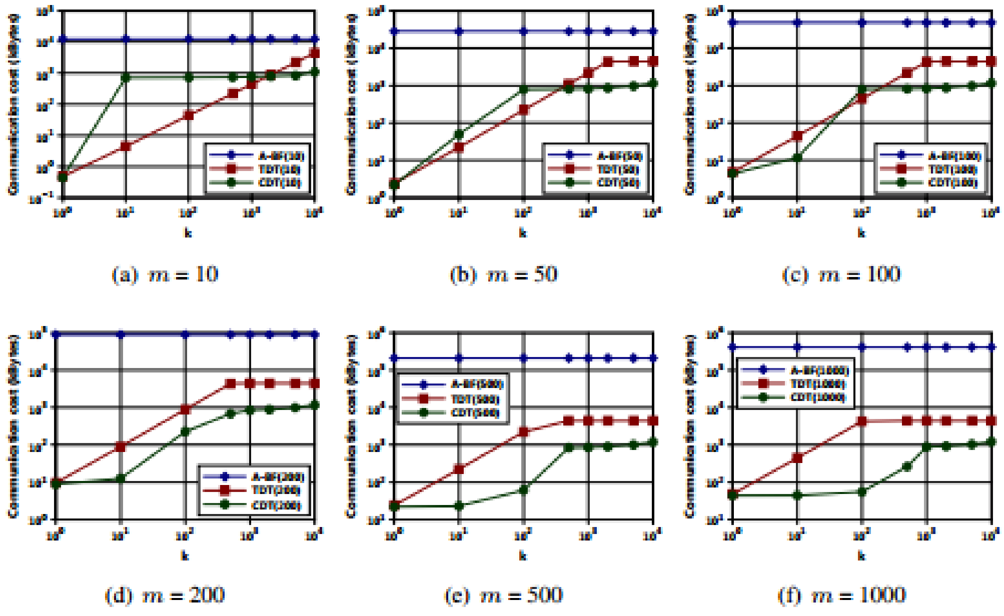

Figure 1. Communication costs of the proposed algorithms and *A-BF* with different values of *m* and *N*

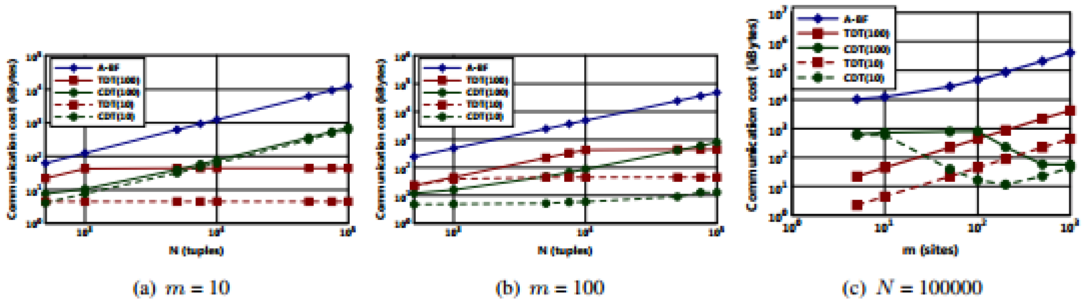

Figure 2. Communication costs with different values of *N* and different values of *m*

5. CONCLUSION

In this paper a novel framework is proposed for ranking distributed uncertain database. The framework consists of three main layers (*Query*; *Ranking* and *Monitoring*), each one has its own algorithms. The main contribution of this paper is at both *Ranking* and *Monitoring* layers. Two novel algorithms for ranking distributed uncertain database are proposed at *Ranking* layer. Expected rank is the ranking function used in both algorithms. The first algorithm (TDT*k*) utilizes only one communication round with *m* x *k* tuples sent to query site. The second one (CDT*k*) utilizes two communication rounds with only (*k* + *m*) tuples, at most, in the first round. An experimental study is made to test the efficiency and effectiveness of the proposed algorithms against *A-BF* algorithm [26]. From observing the experiments results, it is clear that TDT*k* overcomes CDTk in some situations while CDT*k* overcomes TDT*k* in other situations, and both of them overcome the old one.

REFERENCES

[1] P. Cao and Z. Wang, "*Efficient top-k query calculation in distributed networks*", in Proceedings of the twentythird annual ACM symposium on Principles of distributed computing, ser. PODC '04. New York, NY, USA: ACM, 2004, pp. 206–215.

[2] H. Yu, H.G. Li, P. Wu, D. Agrawal, and A. El Abbadi, "*Efficient processing of distributed top-k queries*", in Proceedings of the 16th international conference on Database and Expert Systems Applications, ser. DEXA'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 65–74.

[3] M. Wu, J. Xu, X. Tang, and W.C. Lee, "Top-k monitoring in wireless sensor networks," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, no. 7, pp. 962–976, July 2007.

[4] P. Andreou, D. Zeinalipour-Yazti, P.K. Chrysanthis, and G. Samaras, "Power efficiency through tuple ranking in wireless sensor network monitoring," *Distrib. Parallel Databases*, vol. 29, no. 1-2, pp. 113–150, February 2011.

[5] Y. Cho, J. Son, and Y.D. Chung, "*Pot: an efficient top-k monitoring method for spatially correlated sensor readings*," in Proceedings of the 5th workshop on Data management for sensor networks, ser. DMSN '08. New York, NY, USA: ACM, 2008, pp. 8–13.

[6] A. Qian, Y. Lu, D. Xiaofeng, L. Zou, and Z. Li, "*Efficient top-k monitoring of abnormality in sensor networks*," in Proceedings of the 2009 Ninth IEEE International Conference on Computer and Information Technology -Volume 02, ser. CIT '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 348–353.

[7] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," J. *Comput. Syst. Sci*., vol. 66, no. 4, pp. 614–656, June 2003.

[8] A. Marian, N. Bruno, and L. Gravano, "Evaluating top-k queries over web-accessible databases," *ACM Trans. Database Syst*., vol. 29, no. 2, pp. 319–362, June 2004.

[9] B. Babcock and C. Olston, "*Distributed top-k monitoring*," in Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ser. SIGMOD '03. New York, NY, USA: ACM, 2003, pp. 28–39.

[10] L. Xiong, S. Chitti, and L. Liu, "*Top-k queries across multiple private databases*," in Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, ser. ICDCS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 145–154.

[11] T. Neumann, M. Bender, S. Michel, R. Schenkel, P. Triantafillou, and G. Weikum, "Distributed top-k aggregation queries at large," *Distrib. Parallel Databases*, vol. 26, no. 1, pp. 3–27, August 2009.

[12] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams", *ACM Trans. Database Syst*., vol. 32, no. 4, pp. 1–32, November 2007.

[13] A. Vlachou, C. Doulkeridis, K. Nørv ˚ ag, and M. Vazirgiannis, "*On efficient top-k query processing in highly distributed environments*", in Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 753–764.

[14] A. Deshpande, C. Guestrin, S.R. Madden, J.M. Hellerstein, and W. Hong, "*Model-driven data acquisition in sensor networks*," in Proceedings of the Thirtieth International Conference on Very Large Data Bases – Volume 30, ser. VLDB '04. VLDB Endowment, 2004, pp. 588–599.

[15] M. Ye, X. Liu, W.C. Lee, and D.L. Lee, "*Probabilistic top-k query processing in distributed sensor networks*," in Proceedings of the 26th IEEE International Conference on Data Engineering. Los Alamitos, CA, USA: IEEE Computer Society, 2010, pp. 585–588.

[16] N.N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases", *VLDB Journal*, vol. 16, no. 4, pp. 523–544, 2007.

[17] C. Re, N. Dalvi, and D. Suciu, "*Efficient top-k query evaluation on probabilistic data*," in Proceedings of the 23th IEEE International Conference on Data Engineering. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 886–895.

[18] T. Calders, C. Garboni, and B. Goethals, "*Efficient pattern mining of uncertain data with sampling*," in Advances in Knowledge Discovery and Data Mining, ser. Lecture Notes in Computer Science, M. Zaki, J. Yu, B. Ravindran, and V. Pudi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6118, pp. 480–487.

[19] C. Li, L. Huang, and L. Tian, "*Efficient building algorithms of decision tree for uniformly distributed uncertain data*." in Seventh International Conference on Natural Computation, ICNC 2011, Y. Ding, H. Wang, N. Xiong, K. Hao, and L. Wang, Eds. IEEE, 2011, pp. 105–108.

[20] C.W. Lin and T.P. Hong, "A new mining approach for uncertain databases using cufp trees," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4084–4093, March 2012.

[21] C.W. Lin, T.P. Hong, Y.-F. Chen, T.C. Lin, and S.T. Pan, "An integrated mffp-tree algorithm for mining global fuzzy rules from distributed databases", *Journal of Universal Computer Science*, vol. 19, no. 4, pp. 521–538, feb 2013.

[22] M.A. Soliman, I.F. Ilyas, and K.C.C. Chang, "*Top-k query processing in uncertain databases*," in Proceedings of the 23th IEEE International Conference on Data Engineering, 2007, pp. 896–905.

[23] M. Hua, J. Pei, W. Zhang, and X. Lin, "*Efficiently answering probabilistic threshold top-k queries on uncertain data*", in Proceedings of the 24th IEEE International Conference on Data Engineering, 2008, pp. 1403–1405.

[24] X. Zhang and J. Chomicki, "Semantics and evaluation of top-k queries in probabilistic databases", *Distributed and Parallel Databases*, vol. 26, no. 1, pp. 67–126, 2009.

[25] J. Jestes, G. Cormode, F. Li, and K. Yi, "Semantics of ranking queries for probabilistic data," *IEEE Trans. Knowl. Data Eng*., vol. 23, no. 12, pp. 1903–1917, December 2011.

[26] F. Li, K. Yi, and J. Jestes, "*Ranking distributed probabilistic data*", in Proceedings of the 2009 ACM SIGMOD nternational Conference on Management of Data, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 361–374.

[27] Y. AbdulAzeem, A. Eldesouky, and H. Ali, "Ranking in uncertain distributed database environments", in *The Seventh International Conference on Computer Engineering and Systems (ICCES)*, 2012, pp. 275–280.