An Efficient Algorithm for K-Rank Queries on Large Uncertain Databases

Abdu Gumaei[1*], Rachid Sammouda[1], and AbdulMalik S. Al-Salman[1]

[1]Department of Computer Science, King Saud University, Riyadh, Saudi Arabia

Summary
Recently, large uncertain databases have attracted much attention in many applications, including data management, data integration, social media and security investigation and so on. K-Rank queries, according to matching scores, are an important tool for exploring large uncertain data sets. Few algorithms have been developed to solve this problem. In spite of these works, developing more efficient algorithm is on demand. The problem can be represented as a model of n tuples consist of m instances, and each query-tuple randomly instantiates into one or more tuples based on a set of multi-alternative instances. In this paper, we present an effective backtracking-based algorithm, called Fast Multi-Objective Optimization (FMOO) algorithm. It is able to find K-Rank queries on uncertain databases with efficient memory usage and time complexity $O(knlogn)$, whereas all existing algorithms run in quadratic space and time complexity. Experimental evaluation on synthetic data with theoretical analysis have been provided to demonstrate the efficiency of the new algorithm.

Key words:
Large uncertain databases, K-Rank queries, dominating vectors, Tuples, Instances.

1. Introduction

In our life, some problems need to be ranked for getting results. In [1], the authors give some scenarios of some applications which need to be ranked such as, "when a search engine searches something on the Internet, it often needs to rank a large number of web pages and return the most relevant ones as the result. When querying a database, there could be many tuples that satisfy a given requirement. These tuples need to be ranked and the most relevant ones returned. In general, when many things satisfy a given requirement, we are interested only in the most relevant ones; in particular, we typically do not care about the ranks of the rest". Recently, large uncertain databases have attracted much attention in many applications, including security investigation in social media, data integration [2], web services [3], data cleaning and query processing [4-11], data management of mobiles and sensors [12, 13] and so on. Some important works based on this topic appeared from time to time, such as probabilistic in databases and semantics [14-18]. However, only few works [3, 19, 20] tried to solve the time complexity of Top K-Rank queries algorithms. In the literature, there is another definition of Top K-Rank queries which is the uncertain k-Ranks query (U-k-Ranks), where each tuple in the result is the most probable tuple to appear at a given rank over all possible attributes. The first work [19] started to investigate top-k queries in uncertain databases with exponential time complexity. After that, in [20], the authors proposed an algorithm to solve U-k-Ranks problem with time complexity $O(kn^2)$. Skoutas et al. [3] proposed an algorithm for finding Top-k dominant web services under multi-criteria matching with time complexity $O(kn^2m^2)$. Even for such works, identifying a correct ranking for the candidate matches with efficient time is not straightforward. In this paper, we proposed a general and effective FMOO algorithm to solve K-Rank queries on large uncertain databases with time complexity $O(knlogn)$ and efficient memory usage. This work is considered the first work which reduces the running time to near linear compared to all previous and existing works. The remaining part of this paper is organized as follows: Section 2 presents the research methodology and the proposed algorithm. Section 3 introduces the time complexity analysis of the proposed algorithm and some comparisons with other algorithms. Experimental evaluation is presented in Section 5. Finally, Section 6 concludes the work of this paper.

2. Methodology

The problem of K-Rank queries based on multi-alternative instances is described in this section. The formal definition of our model is as follows: Let N be the set of all tuples stored in the database and let A and V be the sets of possible instances and values, respectively. Let S be the set of possible instance-value pairs, such that S = {(a, v): a $\in$ A and v $\in$ V}. We define two functions: Instance: S $\rightarrow$ A and Value: S $\rightarrow$ V, which give the instance and value, respectively. When we apply those two functions to a specific tuple t where Instance (t) = a, and Value (t) = v, a Rank-k queries T $\in$ N, is a set of tuples where no two distinct tuples can have the same instance, as in Eq. (1).

$$T = \{\forall t1, t2 \in S, t1 \neq t2 \leftrightarrow Ins\tan ce(t1) \neq Ins\tan ce(t2)\}. \quad (1)$$

Manuscript received April 5, 2017
Manuscript revised April 20, 2017

For every query request, the matching process is done on all tuples stored on the uncertain databases to compute the similarity matrix M, as shown in Fig. 1.

Next step starts by sorting the rows of similarity matrix (M) in descending order using merge sort to get a sorted matrix (SM). To keep track of the new locations of matched instances, we store the old indices of M in a matrix called INDXES, as shown in Fig. 1.

In the final step, we apply the proposed algorithm to obtain the K-rank queries based on multi-alternative instances. The proposed algorithm is also able to retrieve a sub-set of K-Rank tuples using a threshold on the average value of current candidate query vector. The main idea of the proposed algorithm is to use the backtracking technique for efficient memory usage and fast processing.

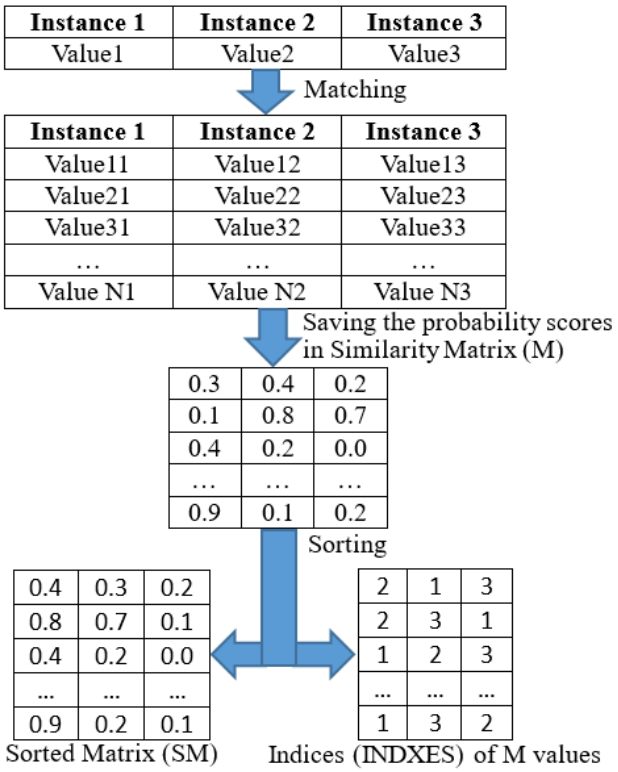

Fig. 1: Matching and sorting processes to obtain sorted and indices matrices

The problem that our algorithm tries to solve can be formulated as a search space problem. The initial state of this space is the first column of matrix (SM), and its weight is computed based on the average of using this column using Eq. (2).

$$avg_1 = \frac{\sum_{i=1}^{N} C_{i,1}}{N} \qquad (2)$$

where $C_{i,1}$ represents the value of row i of the first column in the matrix SM, N is the number of rows in the matrix SM. Expanding states are generated by taking each element in the next column for each row of the matrix SM. The weight value of each expanded state is calculated by the value of that element plus the average value of previous column using Eq. (3).

$$w_{i,j} = \left(\frac{\sum_{i=1}^{N} C_{i,j-1}}{N} \right) + C_{i,j} \qquad (3)$$

where $C_{i,j}$ denotes the value of row i and column j in the matrix SM, $j = 2 \ldots M$, N is the number of rows and M is the number of columns in the matrix SM. In our algorithm, we take the average because it is a closer approximation of the largest column's values. The pseudo code of the proposed algorithm are shown in Fig. 2.

3. Time Complexity Analysis

In general, time complexity is an extremely important issue when the scale of an application grows. Complexity analysis helps us understand and estimate the efficiency of any algorithm. The Big-O notation is used to approximate the time complexity and the worst case performance of an algorithm for a large number of n. In this section, the time complexity of the proposed algorithm is analyzed, as follows. Lines 3-8 of the algorithm cost $O(n)$, lines 11-17 cost $O(nlogn)$ and lines 19-41 cost $O(klogn + kn + kn + knlogn)$. Therefore, total time complexity of the algorithm is $O(knlogn)$. Compared to the other algorithms in literature, the time complexity is reduced, as shown in Table 1.

Table 1: The Time complexity of FMOO proposed algorithm compared to other algorithms in state-of-the-art

Running Time for K-Ranks with multi-alternatives	
Proposed algorithm	$O(knlogn)$
[20]	$O(kn^2)$
[3]	$O(kn^2m^2)$
[19]	$Exponential$

```
/*Input  SM     : the sorted matrix of similarity matrix M
         INDXES: the indices of SM in similarity matrix M
         n,m   : the number of SM rows and the number of SM columns
         K     : the number of desired Ranks based on multi-alternative instances
 Output  kRanksArr: is an array of nxk, each column represent a rank vector contains
         the indices of multi-alternative instances in similarity matrix M */
 1 Begin
 2  accumulator=0;
 3  for i=1 to n do
 4   Begin
 5    C(i)=1;
 6    accumulator=accumulator+SM(i,C(i));
 7    kRanksArr(i,1)=INDXES(i,C(i));
 8   End
 9  Average=accumulator/n;
10  backtrackingAddress=1;
11  for i=1 to n do
12   Begin
13    Weight=Average+SM(i,C(i)+1);
14    Add C(i) to subListArray;
15    ParentID=backtrackingAddress; currentState=i;
16    Insert into MaxHeap Weight, ParentID and currentState;
17   End
18  Add subListArray to mainListArray;
19  for j=2 to K do
20   Begin
21    Delete from MaxHeap maximum Weight with its ParentID and currentState;
22    tempListArray = mainListArray(ParentID);
23    for i=1 to n do C(i)=tempListArray.get(i);
24    C(currentState)=C(currentState)+1;
25    accumulator=0;
26    backtrackingAddress=backtrackingAddress+1;
27    for i=1 to n do
28     Begin
29      accumulator=accumulator+SM(i,C(i));
30      kRanksArr(i,j)=INDXES(i,C(i));
31      Add C(i) to subListArray;
32     End
33    Average=accumulator/n;
34    Add subListArray to mainListArray;
35    for i=1 to n do
36     Begin
37      Weight=Average+SM(i,C(i)+1);
38      ParentID=backtrackingAddress; currentState=i;
39      Insert into MaxHeap Weight, ParentID and currentState;
40     End
41   End
42 End
```

Fig. 2: The pseudo code of FMOO algorithm for finding K-Ranks based on backtracking strategy

4. Experimental Evaluation

In this section, the computational cost is evaluated to prove the correctness of the proposed algorithm. Experimental evaluation is done on synthetic data generated randomly for different number of tuples starting from 100,000 to 1000,000. The other parameters are fixed and summarized in Table 2. As k is a constant, Fig. 3 proves that the running time curve of our algorithm is approximately similar to the curve of $nlogn$ function.

Table 2: The parameters values of experimental evaluation

K-Ranks	Number of Tuples (n)	Number of Instances (m)	Time (ms)
12	100,000	50	1405
12	200,000	50	1647
12	300,000	50	1848
12	400,000	50	2148

12	500,000	50	2610
12	600,000	50	3268
12	700,000	50	3917
12	800,000	50	4859
12	900,000	50	5735
12	1000,000	50	6968

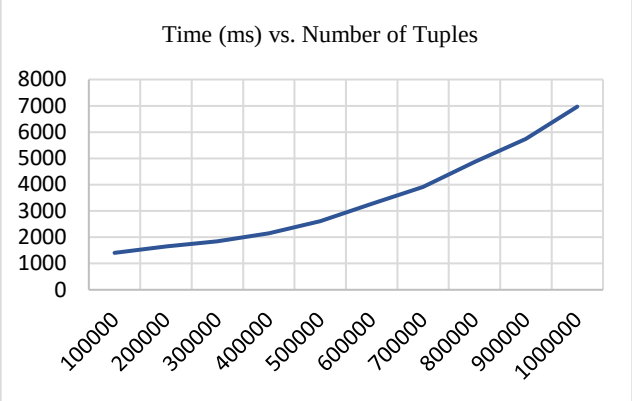

Fig. 3: Running time of the proposed algorithm on large number of tuples

4. Conclusion and future work

In this paper, a backtracking based algorithm, called FMOO algorithm is proposed to solve the problem of *K*-Rank queries on uncertain databases. The FMOO algorithm runs in $O(knlogn)$-time complexity with low memory usage. Compared to other algorithms in the literature review, the time complexity is reduced. This research is considered the first work to solve the *K*-Ranks problem in near polynomial time. The algorithm is also applicable for many applications, including *K*-Ranks of web services, *K*-Ranks of crimes in digital forensics, *K*-Ranks of choices in shopping and many other applications which use the *K*-Ranks of queries on uncertain databases. In the future work, we will apply the FMOO algorithm in different applications and evaluate its efficiency and applicability.

Acknowledgment

This project was funded by the National Plan for Science, Technology and Innovation (MAARIFAH), King Abdulaziz City for Science and Technology, Kingdom of Saudi Arabia, Award no. INF2696-02-12 support.

References

[1] C. Wang, L. Y. Yuan, and J. You. "Top-k ranking for uncertain data," in Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on, vol. 1, pp. 363-368. IEEE, 2010.

[2] A. Halevy, A. Rajaraman, and J. Ordille, "Data integration: the teenage year," in VLDB, 2006.

[3] D. Skoutas, D. Sacharidis, A. V. Kantere, and T. Sellis, "Top-k dominant web services under multi-criteria matching," in Proceedings of the 12th international conference on extending database technology: advances in database technology, pp. 898-909. ACM, 2009.

[4] C. Subramanian, T. Bhuvaneswari, and S. P. Rajagopalan, "Multi Structural Query Engine on a Large Database using Vector Space Approach," IJCSNS, vol.11, no.6, pp.152-159, 2011.

[5] C. V. Neethu, "A Survey of Techniques for Answering Top-k queries," Global Journal of Computer Science and Technology, vol.13, no.2, 2013.

[6] Y. Wang, X. Li, X. Li, and Y. Wang, "A survey of queries over uncertain data," Knowledge and information systems, vol.37, no.3, pp.485-530, 2013.

[7] Y. Lan, S. Niu, J. Guo, and X. Cheng, "Is top-k sufficient for ranking?," in Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pp.1261-1270. ACM, 2013.

[8] J. Zhang, J. Tang, C. Ma, H. Tong, Y. Jing, and J. Li, "Panther: Fast top-k similarity search on large networks," in Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp.1445-1454. ACM, 2015.

[9] J. Sajeev, and V. A. Noorjahan, "Top-K Dominating Queries on Incomplete Data: A Survey," 2016.

[10] P. Wang, B. Wang, and S. Luo, "Top-K Similarity Search for Query-By-Humming," in International Conference on Web-Age Information Management, pp. 198-210. Springer International Publishing, 2016.

[11] H. Dinari, "A Survey on Graph Queries Processing: Techniques and Methods," 2017.

[12] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in SIGMOD, 2003.

[13] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in VLDB, 2004.

[14] S. Abiteboul, P. Kanellakis, and G. Grahne, "On the representation and querying of sets of possible worlds," in SIGMOD, 1987.

[15] D. Barbara, H. Garcia-Molina, and D. Porter, "The management of probabilistic data," IEEE TKDE, vol.4, no.5, pp.487–502, 1992.

[16] N. Fuhr, "A probabilistic framework for vague queries and imprecise information in databases" in VLDB, 1990.

[17] T. Imielinski, and W. Lipski, "Incomplete information in relational databases," J. ACM, vol.31, vol.4, 1984.

[18] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian, "ProbView: a flexible probabilistic database system," ACM TODS, vol.22, no.3, pp.419–469, 1997.

[19] M. A. Soliman, I. F. Ilyas, and K. C. Chang, "Top-K Query Processing in Uncertain Databases," Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2007.

[20] K. Yi, F. Li, G. Kollios, and D. Srivastava, "Efficient processing of top-k queries in uncertain databases with x-relations," IEEE transactions on knowledge and data engineering, vol.20, no.12, pp.1669-1682, 2008.