World Wide Web
DOI 10.1007/s11280-012-0201-5

Efficient top-*k* similarity join processing over multi-valued objects

Wenjie Zhang · Liming Zhan · Ying Zhang · Muhammad Aamir Cheema · Xuemin Lin

Received: 6 September 2012 / Revised: 17 December 2012 / Accepted: 28 December 2012

Abstract The top-*k* similarity joins have been extensively studied and used in a wide spectrum of applications such as information retrieval, decision making, spatial data analysis and data mining. Given two sets of objects $\mathcal{U}$ and $\mathcal{V}$, a top-*k* similarity join returns *k* pairs of most *similar* objects from $\mathcal{U} \times \mathcal{V}$. In the conventional model of top-*k* similarity join processing, an object is usually regarded as a point in a multi-dimensional space and the similarity is measured by some simple distance metrics like Euclidean distance. However, in many applications an object may be described by multiple values (instances) and the conventional model is not applicable since it does not address the distributions of object instances. In this paper, we study top-*k* similarity join over multi-valued objects. We apply two types of quantile based distance measures, ϕ-quantile distance and ϕ-quantile group-base distance, to explore the relative instance distribution among the multiple instances of objects. Efficient and effective techniques to process top-*k* similarity joins over multi-valued objects

Wenjie Zhang was partially supported by ARC DE120102144 and DP120104168.
Ying Zhang was partially supported by ARC DP110104880 and UNSW ECR grant PSE1799.
Muhammad Aamir Cheema was partially supported by ARC DE130101002 and DP130103405.
Xuemin Lin was partially supported by ARC DP0987557, ARC DP110102937, ARC DP120104168, and NSFC61021004.

W. Zhang (✉) · L. Zhan · Y. Zhang · M. A. Cheema · X. Lin
School of Computer Science & Engineering,
University of New South Wales, Sydney, Australia
e-mail: zhangw@cse.unsw.edu.au

L. Zhan
e-mail: zhanl@cse.unsw.edu.au

Y. Zhang
e-mail: yingz@cse.unsw.edu.au

M. A. Cheema
e-mail: macheema@cse.unsw.edu.au

X. Lin
e-mail: lxue@cse.unsw.edu.au

are developed following a filtering-refinement framework. Novel distance, statistic and weight based pruning techniques are proposed. Comprehensive experiments on both real and synthetic datasets demonstrate the efficiency and effectiveness of our techniques.

Keywords Query processing · Joins · Multi-valued objects

1 Introduction

Given two sets of objects (points) $\mathcal{U}$ and $\mathcal{V}$ in a d-dimensional metric space, the top-k similarity join query retrieves k pairs of objects $\mathcal{P}$ from $\mathcal{U} \times \mathcal{V}$ such that the distance between any pair of objects in $\mathcal{P}$ is not greater than the distance of any object pairs in $\mathcal{U} \times \mathcal{V} - \mathcal{P}$. Conventional similarity join query has been extensively studied in various applications including data mining, information retrieval, and location based services [2, 10, 11]. Top-k similarity join, also called *closest pair queries*, has also attracted much research attention [6]. In many applications such as decision making and e-business, an object may be represented by multiple points (instances) in the d-dimensional space, namely *multi-valued* objects [7].

The need of similarity join over multi-valued objects stems from many important applications. In geographic information system (GIS), a group of simple spatial objects may be evaluated as a whole [12, 21]. For instance, to evaluate a community, a real estate development company may model it as a multi-valued object and each instance corresponds to a property with some feature values such as property price, household income, distance to beach, distances to living facilities, etc. A top-k similarity join may be issued to identify the most similar communities from two large cities or from two countries, such that the price fluctuation of one community could be used as a mirror to the management of another one. Similarly, in sports, the performance of a player may be described by her game-to-game statistics in various games. So each player could be represented by a multi-valued object where each instance corresponds to her statistics, such as heights and number of trials in high-jump, in a particular game she attended. A similarity join over two sets of players may help to retrieve players with similar performances. Hence, the successful career path of one player provides a prediction of the success of her counterpart in coming competitions.

Similarity join is also a fundamental and crucial analyzing tool in internet and web information systems. In online shopping systems, it is interesting to retrieve similar pairs of shops or sellers where a shop or seller is modeled by a set of retail items with various features including item type and price range. Here a shop or seller could be modeled as a multi-valued object where each instance corresponds to a retail item. Identifying similar communities from online social networks is another important application of similarity join over multi-valued objects where one community (modeled as a multi-valued object) consists a set of individuals (instances) [19, 24].

While the similarity between two conventional d-dimensional objects only involves two single points, identifying the most similar object pairs among multi-valued object sets involves multiple instances per object. Therefore, it is highly desirable to consider the relative instance distributions among multi-valued objects so that the similar pairs can be effectively retrieved. In this paper, we investigate the problem of similarity join over multi-valued objects in a top-k fashion. That is, we aim to retrieve k pairs of multi-valued objects with the highest level of similarity.

The existing model for handling similarity joins over objects with multiple instances follows the probabilistic semantics on uncertain objects [4, 13, 16] and aims to capture relative instance distribution among objects with multiple instances. Nevertheless, uncertain objects are inherently different than multi-valued objects. Instances of an uncertain object are mutually exclusive which means at most one instance can appear at a particular time, while all the values/instances of a multi-valued object must occur simultaneously at any time. Moreover, as shown in [26], models based on uncertain semantics cannot always capture the relative distributions of multi-valued objects. Take the example in Figure 1. For simplicity we assume multi-valued object U_1 has only one instance with the value (score) of 10, while multi-valued objects V_1 and V_2 both have m instances spread between 9.0 to 9.99 as depicted in Figure 1a. Each instance from the same object takes the same weight. Suppose we want to retrieve the top-1 similarity join result from $\{U_1\}$ and $\{V_1, V_2\}$, namely, retrieve the more similar one from V_1 and V_2 to U_1. Following the possible world semantics, it is easy to verify that both V_1 and V_2 have the same probability, $\frac{1}{2}$, to be the most similar one to U_1 if Euclidean distance is used as the similarity metrics. We permute the distribution in Figure 1a to the distribution in Figure 1b, V_1 and V_2 still have the same probability. This example demonstrates that the probabilistic approaches following the possible world semantics are not able to capture the relative distributions of instances. Another direct solution is to utilize simple aggregates such as average. Nevertheless, such a simple aggregate will have the same problem as pointed above regarding Figure 1.

The example in Figure 1 demonstrates that the existing probabilistic model and simple aggregates may be insensitive to relative distributions of object instances. Quantiles [25] provide a succinct summary of data distributions and is less sensitive to outliers. In this paper, we investigate the top-k similarity join problem over multi-valued objects based on a ϕ-quantile distance ($\phi \in (0, 1]$); for example, median is the 0.5-quantile, maximum is the 1-quantile, minimum is the smallest quantile (note a quantile ϕ is in (0, 1] and cannot be 0). Regarding the above example, 0.5-quantile is based on players' median performance; 1-quantile is to retrieve the top-k similar pairs based on players' worst performance. ϕ-quantile group-base distance, on the other hand, aims at the "best population" (specified by ϕ) regarding the distance of

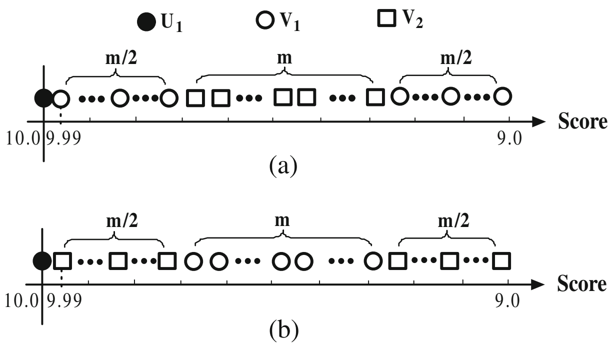


Figure 1 Motivating example

each object pair. Detailed definitions will be provided in Section 2. In this paper, we study the problem of top-k similarity joins over multi-valued objects where the input are two sets of multi-valued objects, and the two types of quantile distances are applied respectively.

Challenges and contributions To the best of our knowledge, this is the first paper to systematically study top-k similarity joins over multi-valued objects. ϕ-quantile distance and ϕ-quantile group-base distance are first used for capturing instance distributions of multi-valued objects in [26]. Zhang et al. [26] studies k-nearest neighbors (KNN) queries over multi-valued objects. Given a multi-valued query object Q and a set of multi-valued objects $\mathcal{U}$, a KNN query retrieves k objects from $\mathcal{U}$ with smallest quantile-based distance to Q. An immediate way to solve our problem can be conducted as follows. For each object $U \in \mathcal{U}$ (or $V \in \mathcal{V}$), we compute its KNN in $\mathcal{V}$ (or $\mathcal{U}$) using the techniques in [26], and then select k most similar pairs based on the union of KNN results. Nevertheless, this involves the computation of KNN for each object in $\mathcal{U}$ (or each object in $\mathcal{V}$). Clearly, not every object in $\mathcal{U}$ (or $\mathcal{V}$) will be involved in the top-k pairs since k is usually much smaller than $min\{|\mathcal{U}|, |\mathcal{V}|\}$. Motivated by this, in this paper, we present a set of novel, efficient, effective pruning techniques to prevent such redundant computation. Our main contributions of the paper can be summarized as follows.

- We formalize the problem of top-k similarity join over multi-valued objects, regarding two types of quantile-based distance metrics.
- Efficient and effective algorithms are developed to compute the top-k similarity join results over two sets of multi-valued objects based on quantile-based distance metrics. Particularly, we propose novel and efficient distance, statistic and weight based pruning techniques to significantly speed up the computation.
- Comprehensive experiments are conducted on both real and synthetic data to demonstrate the efficiency and effectiveness of our techniques. It also demonstrates that the techniques developed in this paper are up to 2 orders of magnitude more efficient than naively applying KNN techniques in [26].

Organization of the paper The rest of the paper is organized as follows. Section 2 formally defines the problem of top-k similarity join over multi-valued objects regarding two types of quantile distance measures, ϕ-quantile distance and ϕ-quantile group-base distance, and provide some necessary background information. In Section 3, we introduce the filtering-refinement framework, as well as the data structures utilized in the paper. Sections 4 and 5 present techniques for top-k similarity join based on ϕ-quantile distance and ϕ-quantile group-base distance, respectively. In Section 6, we report our experiment results. Some extension of the proposed techniques are discussed in Section 7. Related work is summarized in Section 8 in which we present a comprehensive review of techniques on querying multi-valued objects. The is followed by conclusion and future work in Section 9.

2 Background

We present problem definition and necessary preliminaries in this section. For references, notations frequently used in the paper are summarized in Table 1.

Table 1 The summary of notations

Notation	Definition
$\mathcal{U}, \mathcal{V}$	Two sets of objects in the join query
U (V)	Multi-valued object
E	Entry of R-tree
u (v)	Instance of U (V)—a point in d-dimensional space
$w(u)$ ($w(S)$)	(total) weight of u (the set S)
$d(u, v)$	Euclidean distance between u and v
$d^{lo}(E, E')$	Distance lower-bound between E and E'
$d_\phi(U, V)$	ϕ-quantile distance of U and V
$gb\,d_\phi(U, V)$	ϕ-quantile group-base distance of U and V
$U \times V$	Cartesian product of instances from U to V

2.1 Problem definition

Multi-valued object In our problem definition, an instance of an object U is weighted—weight gives the *representativeness* of an instance in U. For instance, in the examples in Section 1, a game statistic of a player may appear multiple times; consequently a normalized weight (the occurrence of an instance over the total occurrences of all instances) may be used to indicate the representativeness of an instance. Note that the total of such weights in U equals 1.

A multi-valued object U is represented as $\{(u_i, w(u_i)) | 1 \leq i \leq m\}$ where u_i is a point in a d-dimensional space, $0 < w(u_i) \leq 1$ $(1 \leq i \leq m)$, and $\sum_{i=1}^{m} w(u_i) = 1$. We use $\mathcal{U}$ and $\mathcal{V}$ to denote two sets of multi-valued objects involved in the join query.

Below we define the ϕ-quantile distance and ϕ-quantile group-base distance between two multi-valued objects.

Quantile Given a collection S of m elements, each element s_i has a weight $w(s_i)$ where $0 < w(s_i) \leq 1$ and $\sum_{i=1}^{m} w(s_i) = 1$. Let S be sorted increasingly on a search key f—a function; that is, $f(s_i) \leq f(s_j)$ if $i < j$.

Definition 1 (ϕ-quantile of S) Given a ϕ $(0 < \phi \leq 1)$, the ϕ-quantile S_ϕ of S is the **first** element s_i in the **sorted** S on the search key such that $\sum_{j=1}^{i} w(s_j) \geq \phi$.

ϕ-quantile distance For two given objects U and V, there are totally $(|U| \times |V|)$ pairs of instances in $U \times V$ where each pair (u_i, v_j) $(u_i \in U$ and $v_j \in V)$ has the weight $w(u_i) \times w(v_j)$, namely $w(u_i, v_j)$. Clearly, $\sum_{u_i \in U, v_j \in V} w(u_i) \times w(v_j) = 1$. The *Euclidean distance* $d(u_i, v_j)$ [1] between u_i and v_j is called the distance of (u_i, v_j). Let $U \times V = \{((u_i, v_j), w(u_i, v_j)) \mid u_i \in U \;\&\; v_j \in V\}$.

Definition 2 (ϕ-quantile distance of U and V) Given a $\phi \in (0, 1]$, let $U \times V$ be sorted increasingly on the search key—the distance $d(u_i, v_j)$ of each element (u_i, v_j). Then, the distance of the ϕ-quantile of $U \times V$ is called the *ϕ-quantile distance* of $U \times V$, denoted by $d_\phi(U, V)$.

[1] Note that our techniques developed in this paper are based on Euclidean distance; nevertheless they can be immediately extended to cover other distance metrics.

Definition 2 states that if (u, v) is the ϕ-quantile of $U \times V$ (i.e., $(U \times V)_\phi = (u, v)$) then $d(u, v)$ is $d_\phi(U, V)$.

Example 1 Regarding the example in Figure 2, $|U| = 3$ and $|V| = 2$. Assume that $w(u_1) = \frac{1}{2}$, $w(u_2) = w(u_3) = \frac{1}{4}$; $w(v_1) = w(v_2) = \frac{1}{2}$. Consequently, $U \times V$ consists of the following six pairs sorted on their distances increasingly:

$$U \times V = \left\{\left((u_2, v_1), \frac{1}{8}\right), \left((u_3, v_1), \frac{1}{8}\right), \left((u_3, v_2), \frac{1}{8}\right), \left((u_1, v_1), \frac{1}{4}\right), \left((u_2, v_2), \frac{1}{8}\right), \left((u_1, v_2), \frac{1}{4}\right)\right\}.$$

The 0.2-quantile distance $d_{0.2}(U, V)$ of U and V is $d(u_3, v_1)$, $d_{0.5}(U, V)$ is $d(u_1, v_1)$, $d_{0.6}(U, V)$ is also $d(u_1, v_1)$.

Below we introduce ϕ-quantile group-base distance measure, which is defined based on the *top*/*best* quantile-population of S.

Definition 3 (ϕ-quantile population of S) Given a S and a $\phi \in (0, 1]$, a ϕ-quantile population $S_{\phi,P}$ of S is a sub-collection S' of S such that the total weights of the elements in S' is not smaller than ϕ and removing any element from S' makes the total weights in the remaining sub-collection smaller than ϕ.

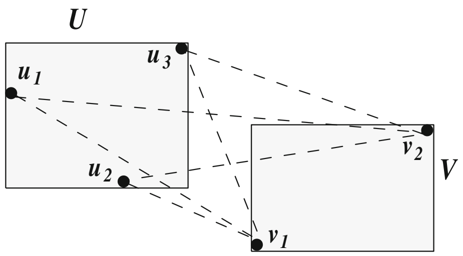
Definition 4 (ϕ-quantile group-base distance) Given a $\phi \in (0, 1]$, the ϕ-quantile group-base distance of U and V is the minimum total weighted distance among ϕ-quantile populations of $U \times V$; that is, the minimum value of $\sum_{(u,v) \in S'} w(u)w(v)d(u, v)$ with the constraint that S' is a ϕ-quantile population of $U \times V$.

The ϕ-quantile group-base distance between U and V is denoted as $gbd_\phi(U, V)$.

Example 2 Regarding Example 1, let $\phi = 0.5$. $gbd_{0.5}(U, V) = \frac{1}{8}d(u_2, v_1) + \frac{1}{8}d(u_3, v_1) + \frac{1}{8}d(u_3, v_2) + \frac{1}{8}d(u_2, v_2)$ instead of $\frac{1}{8}d(u_2, v_1) + \frac{1}{8}d(u_3, v_1) + \frac{1}{8}d(u_3, v_2) + \frac{1}{4}d(u_1, v_1)$.
In fact, there are several 0.5-quantile populations of $U \times V$, including $\{((u_3, v_1), \frac{1}{8}), ((u_2, v_2), \frac{1}{8}), ((u_1, v_1), \frac{1}{4})\}$, $\{((u_2, v_1), \frac{1}{8}), ((u_2, v_2), \frac{1}{8}), ((u_1, v_1), \frac{1}{4})\}$, etc.

Note that for a $\phi \in (0, 1]$, Example 2 shows that $gbd_\phi(Q, U)$ is not always defined on the set of the "consecutive" smallest distances. In Example 2, $\{((u_2, v_1), \frac{1}{8})$, $((u_3, v_1), \frac{1}{8})$, $((u_3, v_2), \frac{1}{8})$, $((u_1, v_1), \frac{1}{4})\}$ is not even a 0.5-quantile population of $U \times V$. In fact, we will show in Section 5 that the computation of $gbd_\phi(Q, U)$ is NP-hard.

Figure 2 Distances between two multi-valued objects



2.1.1 Problem statement

ϕ-quantile top-k similarity join Given a $\phi \in (0, 1]$, two sets of multi-valued objects $\mathcal{U}$ and $\mathcal{V}$ in the d-dimensional space, a ϕ-quantile top-k similarity join retrieves k pairs of objects $\mathcal{P}$ from $\mathcal{U} \times \mathcal{V}$ such that for each object pair (U, V) from $\mathcal{P}$, its ϕ-quantile distance $d_\phi(U, V)$ is no greater than the ϕ-quantile distance of object pairs from $\mathcal{U} \times \mathcal{V} - \mathcal{P}$.

ϕ-quantile group-base Top-k similarity join Given a $\phi \in (0, 1]$, two sets of multi-valued objects $\mathcal{U}$ and $\mathcal{V}$ in the d-dimensional space, a ϕ-quantile group-base top-k similarity join retrieves k pairs of objects $\mathcal{P}$ from $\mathcal{U} \times \mathcal{V}$ such that for each object pair $(U, V) \in \mathcal{P}$, its ϕ-quantile group-base distance $gbd_\phi(U, V)$ is no greater than the ϕ-quantile group-base distance of object pairs from $\mathcal{U} \times \mathcal{V} - \mathcal{P}$.

2.2 Preliminaries

ϕ-quantile distance computation Given a collection S of m elements, each element s_i has a weight $w(s_i)$ where $0 < w(s_i) \leq 1$ and $\sum_{i=1}^{m} w(s_i) \leq 1$. A naive way to compute the ϕ-quantile is to firstly sort S regarding a given search key f, and then scan the sorted list to obtain the ϕ-quantile of S. Clearly, the naive algorithm runs in $O(m \log m)$.

In [5], an efficient and effective partitioning technique PARTITIONING (S) is proposed to find an element $s \in S$ to divide S into two sub-collections S_1 and S_2 with the following properties:

1. for each $s' \in S_1$, $f(s') \leq f(s)$; and for each $s' \in S_2$, $f(s') \geq f(s)$.
2. $|S_1| \geq \frac{3}{10}m - 6$ and $|S_2| \geq \frac{3}{10}m - 6$.

Using the partitioning technique, in Algorithm 1 we present an iteration-based algorithm to compute a ϕ-quantile when S is not sorted.

Algorithm 1 QUANTILE (S, ϕ)

```
Input  : $S$: a collection of $m$ elements; $\phi$: $0 < \phi \leq \sum_{i=1}^{m} w(s_i)$; $f$: specify a search key;
Output : $\phi$-quantile of $S$
1  $(s, S_1, S_2) \longleftarrow$ PARTITIONING (S);
2  if $\phi \leq w(S_1)$ then
3      call QUANTILE $(S_1, \phi)$;
4  end if
5  else
6      if $\phi > w(S_1) + w(s)$ then
7          call QUANTILE $(S_2, \phi - w(S_1) - w(s))$;
8      end if
9      else
10         return $s$;
11     end if
12 end if
```

In Algorithm 1, $w(S_1)$ denotes the total weights of the elements in S_1. When S has only one element, $S_1 = S_2 = \emptyset$. It is shown in [5] that the time complexity of PARTITIONING (S) is linear—$O(|S|)$. Consequently, each iteration runs in linear time regarding the current sub-collection size. Recall the property 2 above in PARTITIONING (S). It is immediate that the sizes of sub-collections involved in the iterations in Algorithm 1 are exponentially reduced—at the ith iteration bounded by $((\frac{7}{10})^{i-1}m + c)$ where c is a constant; consequently, the time complexity of Algorithm 1 is **linear**—$O(m)$. The correctness of Algorithm 1 immediately follows from the property 1 of PARTITIONING (S).

Regarding two multi-valued objects U and V, there are totally $|U| \times |V|$ instance pairs. Directly applying the partition based algorithm, computing ϕ-quantile distance between U and V takes $O(|U| \times |V|)$. In [26], instances inside one multi-valued object are indexed by an R-tree. Based on the R-tree, pruning techniques are proposed to discard instance pairs which are guaranteed not to be the ϕ-quantile of $U \times V$. In this paper, we use the pruning techniques enhanced, partition based, linear time complexity algorithm in [26] as a black box in computing ϕ-quantile distance between two multi-valued objects.

ϕ-quantile group-base distance computation It is proved in [26] that ϕ-quantile group-base distance computation is an NP-hard problem. We adopt the approximate algorithm in [8] while applying it to computing ϕ-quantile group-base distance. Let $approxgbd_\phi(U, V)$ denote the group distance output by the approximation algorithm, the following theorem is shown in [8].

Theorem 1 $1 \leq \frac{approxgbd_\phi(U,V)}{gbd_{phi}(U,V)} \leq 2$

The approximation algorithm runs in $O(m \log m)$ where m is number of instance pairs in $U \times V$. In our experiment, it shows that the approximation algorithm is very accurate in practice.

Conventional Top-k similarity joins As the quantile distance computation between two objects is very expensive with the presence of multiple instances, in this paper, we will apply an R-tree index based top-k similarity join algorithm to facilitate the prevention of computing quantile distances between unpromising pairs of multi-valued objects. In [6], several algorithms are proposed using R-tree based indexes including exhaustive algorithm, recursive algorithm and Heap algorithm. Among all the techniques presented in [6], Heap algorithm demonstrates a better performance in most experiment settings. The priority query based algorithm in [10] is quite similar to Heap algorithm except that Heap algorithm performs a simple pruning before inserting an entry pair into the heap. We adopt the Heap algorithm and develop novel pruning techniques to speed up the computation. Note that our pruning techniques are general enough to be plugged into any R-tree based algorithm for computing conventional top-k similarity joins.

3 Framework

Our techniques for solving the top-k similarity join based on both ϕ-quantile distance and ϕ-quantile group-base distance follow a standard seeding-filtering-refinement framework outlined in Algorithm 2.

Algorithm 2 Framework

- **Phase 1 - Seeding:** Compute the ϕ-quantile distance (ϕ-quantile group-base distance) for each of the k chosen object pairs from $\mathcal{U} \times \mathcal{V}$.
- **Phase 2 - Filtering:** Discard unpromising object pairs from $\mathcal{U} \times \mathcal{V}$.
- **Phase 3 - Refinement:** Determine the final solution for ϕ-quantile top-k similarity join (ϕ-quantile group-base top-k similarity join).

In the seeding phase, we choose k object pairs and compute their ϕ-quantile distances (ϕ-quantile group-base distances), using the techniques introduced in Section 2.2. Let λ_k be the maximal of these k ϕ-quantile distances (ϕ-quantile group-base distances), in the filtering phase, λ_k could be used to prune unpromising object pairs and iteratively updated if necessary. Any k object pairs from $\mathcal{U} \times \mathcal{V}$ could be chosen to compute the ϕ-quantile distance (ϕ-quantile group-base distances) in the seeding phase. Obviously, similar object pairs will lead to smaller λ_k values; and hence better pruning power in the filtering phase. In our framework, to select k object pairs, we first use the mean $\mu(U)$ of the multiple instances for each multi-valued object U from the two given datasets to represent U. $\mu(U) = \sum_{i=1}^{m} w(u_i) \times u_i$ where m is the number of instances in U. Clearly $\mu(U)$ is also in the d-dimensional space. Thus the top-k similarity join is converted to join over conventional datasets where each object is a single point in the multi-dimensional space, and we could apply the existing algorithms [6] to obtain the k most similar pairs from the two (single-valued) datasets. The corresponding k multi-valued object pairs from $\mathcal{U}$ and $\mathcal{V}$ are then chosen to compute the ϕ-quantile distances (ϕ-quantile group-base distances). At this point, we obtain a distance threshold λ_k which will be used in the filtering phase.

3.1 Data structures

In our techniques, we use aggregate R-trees [20] to index the local instances of each multi-valued object in $\mathcal{U} \cup \mathcal{V}$, and use two statistic information enhanced R-trees (named sR-trees) to globally index the minimum bounding boxes (MBBs) of objects in $\mathcal{U}$ and $\mathcal{V}$, respectively. The local aR-trees and global sR-trees are built to facilitate our filtering techniques.

Local aR-trees For each multi-valued object $U \in \mathcal{U} \cup \mathcal{V}$, a *local* aR-tree [20] is built to organize its multiple instances. The aggregate information kept on each intermediate entry is the sum of weights of instances indexed by the entry. Namely, for every intermediate entry E in the local aR-tree, we record the weight of E as the sum of weights (total weights) of instances having E as an ancestor.

Global sR-trees We maintain two R-trees on the MBBs of multiple instances of objects in $\mathcal{U}$ and $\mathcal{V}$, respectively. That is, for each object in $\mathcal{U}$, we first obtain the MBB of its multiple instances. Then we build an R-tree on these MBBs. This R-tree is called the *global* R-tree of $\mathcal{U}$. Similarly we build the *global* R-tree for $\mathcal{V}$. Note in a global R-tree, each leaf (data) entry is an MBB of an object.

Suppose an object U has m instances in the d-dimensional space, $u_1, u_2, ..., u_m$ with the weights $w(u_1), w(u_2), ..., w(u_m)$, respectively.

Definition 5 (Mean μ) The mean of U, denoted by $\mu(U)$, is $\sum_{i=1}^{m} w(u_i) \times u_i$.

Note that $\mu(U)$ is in the d-dimensional space. For $1 \leq i \leq d$, $\mu_i(U)$ denotes the i-th coordinate of $\mu(U)$.

Definition 6 (Variance σ^2) For $1 \leq i \leq d$, $\sigma^2(U) = \sum_{j=1}^{m} w(u_j)(u_{j,i} - \mu_i(U))^2$ where each $u_{j,i}$ denotes the i-th coordinate value of u_j.

In each of the leaf (data) entry of the global R-tree, besides the MBB information of each object, we also keep the above statistic information. And the global R-tree is called a statistic R-tree, denoted by sR-tree. Remind that two sR-tree are built for the multi-valued object sets $\mathcal{U}$ and $\mathcal{V}$, respectively.

4 ϕ-quantile top-k similarity join

We present our techniques for ϕ-quantile top-k similarity join for a given $\phi \in (0, 1]$ in this section. We first present novel distance, statistic and weight based pruning techniques. Then, we integrate the proposed pruning techniques into the overall join algorithm based on the Heap Algorithm in [6].

4.1 Pruning techniques

When introducing the pruning techniques, we assume that we have an entry pair (E_U, E_V) from the join processing where E_U (E_V) is an entry from the global sR-tree of $\mathcal{U}$ $(\mathcal{V})$. E_U (E_V) could be either intermediate or leaf (data) entry. The way to access entries from the two global sR-trees will be introduced in Section 4.2.

Distance based pruning The first pruning rule is based on the distance between two entries in the join processing obtained from intermediate or leaf entries of two global sR-trees.

Pruning Rule 1 Let $d^{lo}(E_U, E_V)$ denote the minimum distance between the MBBs of two entries E_U and E_V. If $d^{lo}(E_U, E_V) \geq \lambda_k$, then (E_U, E_V) can be pruned, namely, all entry pairs in $E_U \times E_V$ can be pruned.

Complexity Computing the minimum distance between two MBBs takes $O(d)$ time. The complexity of Pruning Rule 1 is constant once d is fixed.

Statistic based pruning The second pruning technique utilizes the statistic information kept in the global sR-tree, as introduced in Section 3. The main idea is based on the current distance threshold λ_k, to derive a value α such that the α-quantile distance between an object pair (U, V) is not smaller than λ_k. If $\alpha < \phi$, we can safely prune (U, V). We first introduce the *Cantelli's inequality* [18] which is employed in Pruning Rule 2.

Let $\delta(x, y)$ be $\frac{1}{1+\frac{x^2}{y^2}}$ if $y \neq 0$, 1 if $x = 0$ and $y = 0$, and 0 if $x \neq 0$ and $y = 0$.

Theorem 2 (Cantelli's Inequality [18]) *Suppose that t is a random variable in 1-dimensional space with mean $\mu(t)$ and variance $\sigma^2(t)$, $Prob\,(t - \mu(t) \geq a) \leq \delta(a, \sigma(t))$ for any $a \geq 0$, where $Prob\,(t - \mu(t) \geq a)$ denotes the probability of $t - \mu(t) \geq a$.*

Note that Theorem 2 extends the original Cantelli's Inequality [18] to cover the case when $\sigma = 0$ and/or $a = 0$. The following theorem is proved in [15] and provides an upper-bound for $Prob\,(t \leq b)$ when $b \leq \mu$.

Theorem 3 *Assume that* $0 \leq b \leq \mu(t)$. *Then,* $Prob\,(t \leq b) \leq \delta(\mu(t) - b\,, \sigma(t))$.

Proof Let $t' = 2\mu(t) - t$. It can be immediately verified that $\sigma^2(t') = \sigma^2(t)$ and $\mu(t) = \mu(t')$. Applying Cantelli's Inequality on t', the theorem holds. □

Now we generalize the above observations into our statistic based pruning rule. As shown in Figure 3, for two object entries (U, V) stored in the leaf/data entries of global sR-tree of $\mathcal{U}$ and $\mathcal{V}$, along the i-th dimension ($1 \leq i \leq d$), e.g., the horizontal dimension in Figure 3, we locate two lines m and n vertical to the i-th dimension and with distance λ_k between m and n. Denote U_i (V_i) as the coordinate value of U (V) along the i-th dimension. The line $U_i = m$ ($V_i = n$) divides the MBB of U (V) into two parts, denoted as U_1 and U_2 (V_1 and V_2), as shown in Figure 3. Assume $\mu_i(U) < \mu_i(V)$. Remind that λ_k is the current distance threshold.

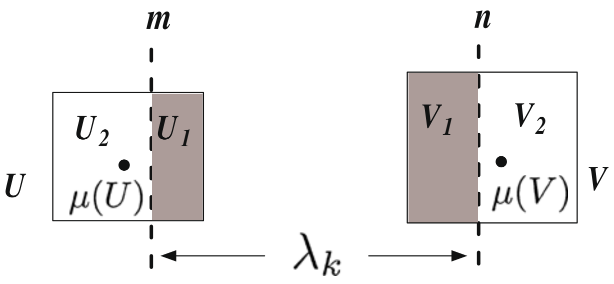
The intuition of the statistic based pruning technique is as follows: along each dimension i, based on Theorem 3, we derive an upper bound of the sum of weights in the shaded areas of the MBBs of U_1 and V_1, respectively, denoted as $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$. Clearly, we can claim that instance pairs from $U_2 \times V_2$ can not have distance smaller than λ_k. Denote the sum of weights in U_2 and V_2 as $W_i(U_2)$ and $W_i(V_2)$, respectively. Obviously, $W_i(U_2) \geq 1 - W_i^{up}(U_1)$, and $W_i(V_2) \geq 1 - W_i^{up}(V_1)$. Thus, using $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$, we can identify a value α such that the α-quantile distance between U and V is not smaller than λ_k. Next we present the monotonic property of quantile distance.

Theorem 4 (Monotonicity of quantile distance) *Given two multi-valued objects U and V, $\alpha, \phi \in (0, 1]$, if $\alpha < \phi$, then $d_\alpha(U, V) \leq d_\phi(U, V)$.*

Proof The theorem immediately holds based on the definition of quantile distance in Definition 2.

Based on Theorem 4, once we identify the value α such that the α-quantile distance between U and V is larger than λ_k, if $\alpha < \phi$, then we can claim the ϕ-quantile distance between U and V cannot be smaller than λ_k. In this way (U, V) can

Figure 3 Statistic based pruning



be pruned based on the statistic information kept in the global sR-tree only without accessing the local aR-trees of U and V.

Pruning Rule 2 Given an object pair (U, V) $(U \in \mathcal{U}, V \in \mathcal{V})$. For a dimension i $(1 \le i \le d)$, without lose of generality, assume $\mu_i(U) < \mu_i(V)$. If $1 - (1 - \delta(m - \mu_i(U), \sigma_i(U))) \times (1 - \delta(\mu_i(V) - n, \sigma_i(U))) < \phi$, (U, V) can be pruned.

Proof For the i-th $(1 \le i \le d)$ dimension, based on Theorem 3, we obtain the upper bound of the sum of weight of instances in the shaded area U_1 of the MBB of U as $W_i^{up}(U_1) = Prob(U_i \ge m) \le \delta(m - \mu_i(U), \sigma_i(U))$. Similarly we get $W_i^{up}(V_1) = Prob(V_i \le n) \le \delta(\mu_i(V) - n, \sigma_i(V))$. Since the instance pairs from $U_2 \times V_2$ cannot have distance smaller than λ_k, we have $\alpha \le 1 - (1 - W_i^{up}(U_1)) \times (1 - W_i^{up}(V_1)) \le 1 - (1 - \delta(m - \mu_i(U), \sigma_i(U))) \times (1 - \delta(\mu_i(V) - n, \sigma_i(U)))$. Together with Theorem 4, the pruning rule is correct. □

Once we obtain an object pair (U, V) from the join processing, we apply Pruning Rule 2 based on the statistic information kept in the global sR-trees before accessing the local aR-trees of U and V. If we encounter a dimension i such that $1 - (1 - W_i^{up}(U_1)) \times (1 - W_i^{up}(V_1)) < \phi$, the pruning stops and the object pair (U, V) is discarded. As shown in Figure 3, after selecting line m along the i-th dimension of U, line n for V is also fixed regarding the current λ_k. We apply the equality principle in determining the position of m and n; namely, the center of m and n is the same as the center of $\mu_i(U)$ and $\mu_i(V)$. Based on Theorem 3, we obtain $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$ in constant time.

Complexity If $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$ are derived based on Theorem 3, the time complexity of Pruning Rule 2 is $O(d)$.

Weight based pruning The following pruning rule incorporates both weight and distance information. The instances of a multi-valued object are investigated by accessing the local aR-trees. Consider an object entry pair (U, V). If (U, V) is not pruned by Pruning Rule 1 and 2, we explore the instances information of the objects by accessing their local aR-trees. We traverse the local aR-trees of two objects U and V synchronously. At level i, we *trim* object V using the current distance threshold λ_k, and retain only the entries in V with minimum distance to U not larger than λ_k. We record the entries as $\gamma_{V,i}$. Formally, $\gamma_{V,i} = \{E \in L_i(V), d^{lo}(U, E) \le \lambda_k\}$, where $L_i(V)$ denotes all remaining entries (i.e., not trimed in higher levels) in the local aR-tree of V at the i-th level. Similarly, we obtain $\gamma_{U,i}$. If the multiplication of the weights of $\gamma_{V,i}$ and $\gamma_{U,i}$ is smaller than ϕ, the object pair (U, V) can be pruned as the ϕ-quantile distance between U and V must be larger than λ_k.

Pruning Rule 3 If $\sum_{e \in \gamma_{U,i}} W(e) \times \sum_{e \in \gamma_{V,i}} W(e) < \phi$, the object pair (U, V) can be discarded.

Proof From the definition of ϕ-quantile distance, it is immediate that if $\sum_{e \in \gamma_{U,i}} W(e) \times \sum_{e \in \gamma_{V,i}} W(e) < \phi$, then $d_\phi(U, V) > \lambda_k$. □

Example 3 As shown in Figure 4, at the i-th level, the local aR-tree of object U has two entries U_1 and U_2, local aR-tree of V also has two entries V_1 and V_2. The current

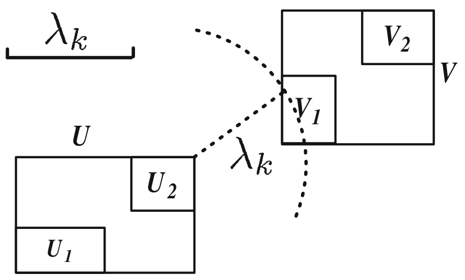


Figure 4 Weight based pruning

threshold λ_k is as illustrated. Using λ_k, we *trim* the MBB of V and only entry V_1 has minimum distance to U smaller than λ_k; thus, $\gamma_{V,i} = \{V_1\}$. Similarly, $\gamma_{U,i} = \{U_2\}$. If $W(U_2) \times W(V_1) < \phi$, the object pair (U, V) could be pruned.

By applying Pruning Rule 3, we can avoid accessing all instance pairs of $U \times V$, and seek to stop on intermediate levels of the local aR-trees of U and V. Note the traversal of two aR-trees is in a synchronous fashion and level-by-level from the root node. If one aR-tree reaches leaf nodes first, it stays in leaf level while the other one keeps traversing till its leaf level. As a by-product, if (U, V) cannot be pruned using Pruning Rule 3, we call the ϕ-quantile distance computation algorithm in [26] with the instance pairs from $\gamma_{U,i} \times \gamma_{V,i}$ only where i is the leaf (instance) level. Clearly, the algorithm still outputs correct ϕ-quantile distance as the distance of the pruned instance pairs are larger than λ_k based on the definition of $\gamma_{U,i}$ and $\gamma_{V,i}$ for level i.

An exceptional case of Pruning Rule 3 is that we obtain an entry pair (E_U, E_V) from the join processing, one is an object entry while the other is an intermediate entry. Assume E_U is the object entry of U and E_V is the intermediate entry. Pruning Rule 3 could still be applied to (U, E_V) with the following modifications: 1) We access the local aR-tree of U only and at each level i, record $\gamma_{U,i}$ as the entries in U with minimum distance to E_V not larger than λ_k; 2) if $\sum_{e \in \gamma_{U,i}} W(e) < \phi$, the entry pair (U, E_V) could be pruned. Namely, the object pair of U and any object indexed in E_V must have a ϕ-quantile distance greater than λ_k.

Complexity Assume the average number of entries at level i of the local aR-trees of multi-valued objects is N_i, then clearly the complexity of Pruning Rule 3 is $O(N_i)$ at each level. The worst case complexity of using Pruning Rule 3 is $O(|U| \times |V|)$, namely no entries are pruned at intermediate entries and we need to access all instance pairs. However, in practice, as shown in Section 6, Pruning Rule 3 is very effective and saves CPU costs significantly. Note that in Pruning Rule 3 we trim the entries at each level of local aR-trees of U and V using λ_k instead of considering the combination of all pairs of entries at each level. This is because trim based pruning is more efficient compared with combining all pairs (time complexity $O(N_i^2)$) and also trim based pruning is very effective in practice.

4.2 Overall join algorithm

The join algorithm used in this paper is adopted from the Heap Algorithm in [6] as it is both efficient and easy to implement in real applications. We adjust the algorithm to deal with multi-valued objects. Given $\phi \in (0, 1]$, two multi-valued objects sets $\mathcal{U}$

and $\mathcal{V}$, Algorithm 3 illustrates the top-k similarity join processing. A minheap H is maintained according to the minimum distance between two entry pairs of the two global R-trees $R_{\mathcal{U}}$ and $R_{\mathcal{V}}$ indexing $\mathcal{U}$ and $\mathcal{V}$, respectively. H is initialized with the pair of root nodes of $R_{\mathcal{U}}$ and $R_{\mathcal{V}}$.

Algorithm 3 Top-k similarity join processing

```
   Input  : $R_{\mathcal{U}}$, $R_{\mathcal{V}}$, $k$, $\phi$
   Output : $k$ object pairs from $\mathcal{U} \times \mathcal{V}$ with smallest $\phi$-quantile distances
1  $H$ = (root($R_{\mathcal{U}}$), root($R_{\mathcal{U}}$)) if not PRUNED1(root($R_{\mathcal{U}}$), root($R_{\mathcal{U}}$));
2  while $H$ is not empty do
3      $(E_U, E_V) = H$.top();
4      $H$. pop();
5      if $E_U$ and $E_V$ are both intermediate entries then
6          for each children pair $(C_{E_U}, C_{E_V})$ from $E_U \times E_V$ do
7              if not PRUNED1($C_{E_U}, C_{E_V}$) then
8                  insert $(C_{E_U}, C_{E_V})$ into $H$;
9              end if
10         end for
11     end if
12     else if one of $E_U$ and $E_V$ is an object entry then
13         if not PRUNED1($E_U, E_V$) and not PRUNED3($E_U, E_V$) then
14             Lines 6 - 10;
15         end if
16     end if
17     else                                   /* both $E_U$ and $E_V$ are object entries */
18         if not PRUNED1($E_U, E_V$) and not PRUNED2($E_U, E_V$) AND not
           PRUNED3($E_U, E_V$) then
19             Compute $\phi$-quantile distance between $E_U$ and $E_V$;
20             if $d_{\phi}(E_U, E_V) < \lambda_k$ then
21                 Update $\lambda_k$ and current $k$ most similar pairs;
22             end if
23         end if
24     end if
25 end while
```

The algorithm differentiates three cases based on whether the entries are object entries or not. If both are intermediate entries (Line **5**), we expand all the children pairs and insert into heap H the pairs which survive from Pruning Rule 1 (Line **7**). If one of the entries is an intermediate entry and the other is an object entry (Line **12**), Pruning Rule 1 and 3 will be applied first (Line **13**) before expanding the children pairs. We apply all 3 Pruning Rules on object pairs (Line **18**), and if an object pair is survived from pruning, the ϕ-quantile distance is computed; the top-k results and λ_k are updated if necessary. Note that even from the root node pair we only insert entry pairs into H if they are not pruned by Pruning Rule 1, it is still necessary to check Pruning Rule 1 (Lines **13** and **18**) since the distance threshold λ_k dynamically changes.

Correctness Based on the correctness of the 3 pruning rules, it can be immediately shown that Algorithm 3 is correct.

Discussions The techniques proposed in this paper could be immediately extended to support self-join (i.e., we compute top-k similar pairs from one data set $\mathcal{U}$) and threshold base similarity join over multi-valued objects. We omit the details due to space limits.

5 ϕ-quantile group-base top-k similarity join

Our techniques for join processing based on *ϕ-quantile group-base distance* also follow the seeding-filtering-refinement framework in Algorithm 2. In the seeding phase, k object pairs are selected based on the distance between weighted centroid, then corresponding ϕ-quantile group-base distances are computed using the approximation algorithm in [8]. The largest among these k distance values is thus utilized as the distance threshold γ_k.

Below we first present the novel and efficient pruning techniques, followed by the overall join processing algorithm based on ϕ-quantile group-base distance metrics.

5.1 Pruning techniques

We assume that we have an entry pair (E_U, E_V) from the join processing where E_U (E_V) could be either an intermediate entry from the global R-tree of U (V), or a data entry from the local aR-tree of U (V). The following pruning rule is immediate based on the definition of ϕ-quantile group-base distance between two multi-valued objects.

Pruning Rule 4 If $\phi \times d^{lo}(E_U, E_V) \geq \gamma_k$, then (E_U, E_V) can be pruned, namely, all entry pairs in $E_U \times E_V$ can be pruned.

Proof For each object pair (U, V) where E_U is an ancestor of U and E_V is an ancestor of V, the total weight for any ϕ-quantile population S of $U \times V$ is not smaller than ϕ and the distances of all instances pairs in S are not smaller than γ_k. Thus the pruning condition holds. □

Complexity Computing the minimum distances between E_U and E_V takes $O(d)$ time, thus the complexity of Pruning Rule 4 is constant once d is fixed.

Note that given U and V, the instance group with the total weighted distance $gb\,d_\phi(U, V)$ may spread in many different entries of U and V, it is not always possible to trim many entries from the local aR-trees as we do for ϕ-quantile similarity join processing in Pruning Rule 2 and Pruning Rule 3 . The next pruning rule further explores the instances distribution information inside multi-valued objects using the local aR-trees. Before presenting the next pruning technique, we differentiate two cases, 1) both E_U and E_V are data entries, i.e., pointing to a multi-valued object; and 2) one of them is an intermediate entry in the global sR-tree and the other is a data entry. Note that Case 2) occurs when the two global sR-trees are of different heights.

Considering Case 1) first. Since both E_U and E_V are data entries, we load the corresponding local aR-trees, respectively. The pruning rule is conducted in a level-by-level fashion between the two local aR-trees starting from the root node pairs. Let $L_{U,k}$ and $L_{V,k}$ denote all entries at level k of the local aR-tree of U and V, respectively. Clearly there are overall $|L_{U,k}| \times |L_{V,k}|$ entry pairs. We denote these entry pairs as $L_k = \{(E_U, E_V)_1, ... (E_U, E_V)_{|L_{U,k}| \times |L_{V,k}|}\}$. Without lose of generality, we assume these entry pairs are sorted in decreasing order based on the minimal distance between the corresponding two MBRs, namely, $d^L(E_U, E_V)_{i_1} \leq d^L(E_U, E_V)_{i_2}$ if $i_1 < i_2$. Let $(E_U, E_V)_j$ denote the ϕ-quantile of L_k where the search key is the minimum distance of two MBRs in each entry pair and the weight of each pair is the multiplication of the weights of the two entries involved in the pair. The intuition of the following pruning rule is to relax the ϕ-quantile distance between U and V to obtain a lower bound of $gbd_\phi(U, V)$.

Pruning Rule 5 (Case 1) Two Multi-valued objects U and V can be pruned if at a level k of the local aR-trees of U and V,

$$\left(\phi - \sum_{i=1}^{j-1} w((E_U, E_V)_i)\right) d^L(E_U, E_V)_j + \sum_{i=1}^{j-1}(w((E_U, E_V))_i \times d^L(E_U, E_V)_i) \leq \gamma_K$$

Notice that in Pruning Rule 5 (Case 1), if the numbers of instances of U and V are different, it is possible that the two local aR-trees are of different hight. In such scenarios, if one aR-tree reaches the leaf (instance) level first, then the other tree keeps traversing until it also reaches the leaf level.

In Case 2), one of the entry is an intermediate entry from the global sR-tree while the other is a data entry. Without loss of generality, we assume E is an intermediate entry from the sR-tree of object U, the pruning rule is conducted by accessing the local aR-tree of object V. Suppose that $L_k = \{E_i | 1 \leq i \leq l\}$ consists of all entries at the level k of the local aR-tree of object V and assume that L_k is sorted in the increasing order based on $d^L(E, E_i)$; namely, $d^L(E, E_{i_1}) \leq d^L(E, E_{i_2})$ if $i_1 < i_2$. Let E_j denote the ϕ-quantile of L_k according to the search key $d^L(E, E_i)$ and the weight $w(E_i)$ of each entry $E_i \in L_k$.

Pruning Rule 5 (Case 2) The object pairs $E \times V = \{(U, V) | E \text{ is the ancestor of } U\}$ could be pruned if at a level k of the local aR-tree of V,

$$\left(\phi - \sum_{i=1}^{j-1} w(E_i)\right) d^L(E, E_j) + \sum_{i=1}^{j-1}(w(E_i) \times d^L(E, E_i)) \leq \gamma_K$$

Proof We prove Case 1) first and Case 2) could be proved in a similar way. Remind that L_k denotes the sorted entry pairs from U and V at k-th level of the corresponding aR-trees where the sorting is conducted according to the minimal distance of MBRs of entry pairs, and $(E_U, E_V)_j$ denotes the ϕ-quantile of L_k. For entry pairs before $(E_U, E_V)_j$ in L_k, we obtain the weighted distance of minimal MBR distances, $\sum_{i=1}^{j-1}(w((E_U, E_V))_i \times d^L(E_U, E_V)_i)$. Deducting this part of weights in ϕ (i.e., $\phi - \sum_{i=1}^{j-1} w((E_U, E_V)_i)$), the remaining part is weighted by the minimal distance of the ϕ-quantile entry pair $(E_U, E_V)_j$. The sum of these two parts is not smaller than the weighted distance in any ϕ-quantile population of $U \times V$. □

Complexity Remind that $|L_{U,k}|$ and $|L_{V,k}|$ denote the number of entries in the k-th level of the local aR-tree of U and V, respectively. The time complexity for Pruning Rule 5 Case 1) is $O(|L_{U,k}| \times |L_{V,k}| \times \log(|L_{U,k}| \times |L_{V,k}|))$ since we need to sort the entry pairs first. Similarly, the time complexity for Pruning Rule 5 Case 2) is $O(|L_k| log |L_k|)$ where $|L_k|$ refers the number of entries at the k-th level of local aR-tree of V.

5.2 Overall join algorithm

Join processing based on group-base ϕ-quantile distance metrics also follows the Heap Algorithm in [6]. Based on Algorithm 3, the following changes are made.

- Output object pairs with smallest approximate ϕ-quantile group-base distance.
- In Line 7, PRUNED1 is changed to PRUNED4.
- In Line 13, PRUNED 1 and PRUNED 3 are changed to PRUNED 4 and PRUNED 5.
- In Line 18, PRUNED 1 PRUNED 2 and PRUNED 3 are changed to PRUNED 4 and PRUNED 5.
- In Line 19, compute $approxgb\,d_\phi(E_U, E_V)$.
- In Line 20, $d_\phi(E_U, E_V)$ is changed to $approxgb\,d_\phi(E_U, E_V)$.

Accuracy guarantee Due to the hardness of computing ϕ-quantile group based distance, our techniques for ϕ-quantile group-base similarity join yield approximate results with the following accuracy guarantee.

Theorem 5 *For $1 \le i \le k$, assume that (U_i, V_i) denotes the ith most similar object pair in the exact ϕ-quantile group-base similarity join, (U'_i, V'_i) denotes the top-ith most similar object pair returned by our algorithms. Then, $gb\,d_\phi(U_i, V_i) \le approxgb\,d_\phi(U'_i, V'_i) \le 2gb\,d_\phi(U_i, V_i)$. Namely, our algorithm has an approximation ratio of 2.*

Proof Following the proof of Pruning Rule 4 and 5, it is immediate that if an object pair (U, V) is pruned by these two pruning rules, then $gb\,d_\phi(U, V) \ge \gamma_k$. From Theorem 1, it follows that $gb\,d_\phi(U_i, V_i) \le approxgb\,d_\phi(U'_i, V'_i) \le 2gb\,d_\phi(U_i, V_i)$. □

Theorem 5 states that the i-th ϕ-quantile group-base distance output by our algorithm is bounded by $gb\,d_\phi(U'_i, V'_i)$ and $2gb\,d_\phi(U'_i, V'_i)$. Our experimental results show that the algorithm is quite accurate in practice and the error is much smaller.

6 Experiment

We report a thorough performance evaluation on the efficiency of proposed techniques and effectiveness of pruning rules. In particular, we implement and evaluate the following techniques.

Join	: Techniques presented in Section 4 to compute the top-k similarity join based on ϕ-quantile distance ($\phi \in (0, 1]$), with Pruning Rule 1, 2 and 3 applied.
P12	: **Join** algorithm but with Pruning Rule 1 and 2 only.
P1	: **Join** algorithm but with Pruning Rule 1 only.
KNN	: Baseline algorithm for **Join** by using KNN processing over multi-valued objects in [26]. For each object $U \in \mathcal{U}$, we compute its KNN in $\mathcal{V}$ based on ϕ-quantile distance, and then select k most similar pairs based on the union of KNN results.

G-Join	: Techniques presented in Section 5 to compute the top-k similarity join based on group-base ϕ-quantile distance ($\phi \in (0, 1]$), with Pruning Rule 4 and 5 applied.
P4	: **G-Join** algorithm but with Pruning Rule 4 only.
G-KNN	: Baseline algorithm for **G-Join** by using group-base quantile KNN processing over multi-valued objects in [26]. For each object $U \in \mathcal{U}$, we compute its KNN in $\mathcal{V}$ based on ϕ-quantile group-base distance, and then select k most similar pairs based on the union of KNN results.

All algorithms are implemented in C++ and compiled by GNU GCC. Experiments are conducted on PCs with Intel Xeon 2.4 GHz dual CPU and 4 G memory under Debian Linux. Our experiments are conducted on both real and synthetic datasets.

Real dataset is extracted from NBA players' game-by-game statistics (http://www.nba.com), containing 339,721 records of 1,313 players. Each player is treated as a multi-valued object where the statistics (score, assistance, rebound) of a player per game is treated as an instance with the equal weight (normalized).

Synthetic datasets are generated using the methodologies in [1] regarding the following parameters. Dimensionality d varies from 2 to 5 with default value 3. Data domain in each dimension is [0, 1]. Number n of objects varies from 5,000 to 15,000 with default value 5,000. Number m of instances per object follows a uniform distribution in $[1, \mathcal{M}]$ where $\mathcal{M}$ varies from 100 to 800 with the default value 200. The value K varies among 5, 10, 15, 20 and 25 with default value 10. The average length of object MBBs follows a *uniform* distribution spreading over $[0, h]$ where h varies from 0.02 to 0.10 with default value 0.02 (i.e., 2 % of the edge length of the whole data space).

Centers of objects (objects' MBBs) follow either *uniform*, *normal* or *anti-correlated* distribution. Locations of instances in an object follow *uniform* or *normal* distribution. Weights assigned to each instance follow *uniform* or *normal* distribution. Table 2 summarizes the parameters used in our experiment where the default values are in **bold** font.

6.1 Overall performance

Figure 5 reports the results of the evaluation on processing time on **Join**, **G-Join** and their corresponding baseline algorithms **KNN**, **G-KNN** on both synthetic and real data. As shown, **Join** and **G-Join** are up to 3 orders of magnitude more efficient than

Table 2 Parameter values

Dimensionality d	2, **3**, 4, 5
Number of objects n	**5k**, 7.5k, 10k, 12.5k, 15k
Edge length h	**0.02**, 0.04, 0.06, 0.08, 0.10
Number of instances m	100, **200**, 400, 600, 800
K	5, **10**, 15, 20, 25
ϕ	0.1, 0.3, **0.5**, 0.7, 0.9
Object location	Uniform, normal, **anti-correlated**
Weight distribution	Uniform, **normal**

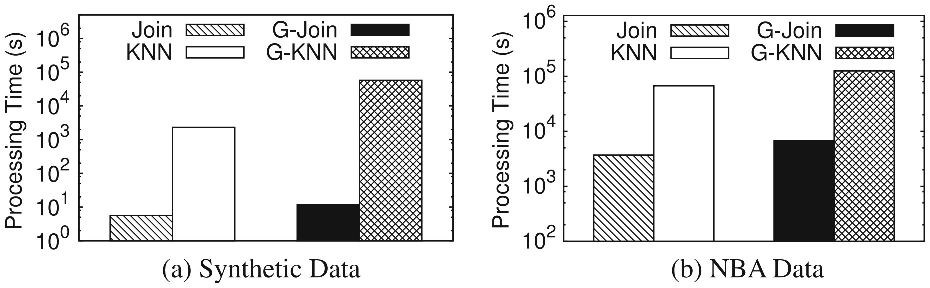


Figure 5 Compare with baseline algorithms

their baseline versions on synthetic data. The improvement is less significant on NBA data because in NBA dataset, objects' MBBs largely overlap so that it is very hard to prune an object. In the rest of the experiments we no longer evaluate the baseline algorithms since their performance is much worse than the techniques proposed in this paper.

6.2 Accuracy

We evaluate the accuracy of **G-Join** in this part. As the ϕ-quantile group-base similarity join is NP-hard and no efficient algorithm exists, we produce the exact solution using exhaustive search which is exponential to the number of instances and very slow. So we conduct a very small scale experiment as follows. Each dataset contains 500 multi-valued objects and each object consists of 4 instances. Other parameters use the default setting in Table 2.

To evaluate the accuracy of **G-Join**, we use two error metrics. The first is the average distance error ratio. For $1 \le i \le K$, $approx(i)$ denotes the group-base ϕ-quantile distance of the top-i-th object pair output by **G-Join**, and $exact(i)$ denotes the group-base distance of the top-i-th object pair in the exact solution.

$$err\_ratio = \frac{\sum_{i=1}^{K} \frac{|approx(i)-exact(i)|}{exact(i)}}{K}$$

The second measure records the "missed" object pairs, namely the object pairs that are missed in the approximate results output by **G-Join**. Let $approx$ denote the K object pairs output by **G-Join**, $exact$ denotes the K object pairs in the exact solution.

$$miss\_ratio = 1 - \frac{|approx \cap exact|}{K}$$

Table 3 Vary objects distribution

	err_ratio	miss_ratio
anti	0.037	0.03
unif	0.029	0.02
norm	0.036	0.02

Table 4 Vary weight distribution

	err_ratio	miss_ratio
unif	0	0
norm	0.037	0.03

We report the results in Table 3 where the object distribution varies, and in Table 4 where the distribution of weights varies. Both demonstrate that **G-Join** is highly accurate and more accurate than the theoretical guarantee in Theorem 5.

6.3 Evaluating effects by different settings

We study the scalability of our algorithms regarding different ϕ values, number of objects, number of instances ($\mathcal{M}$), lengths of MBB edges (h), K, and the dimensionality d.

Figure 6 reports the scalability of **Join** regarding various parameters. It shows that **Join** is not very sensitive to different ϕ and K values, while quite sensitive to other parameters. With a larger number of objects and instances, more objects and instances are involved in the computation, thus incurring higher computation cost. With large h values, the MBBs of objects are more likely to overlap with each other

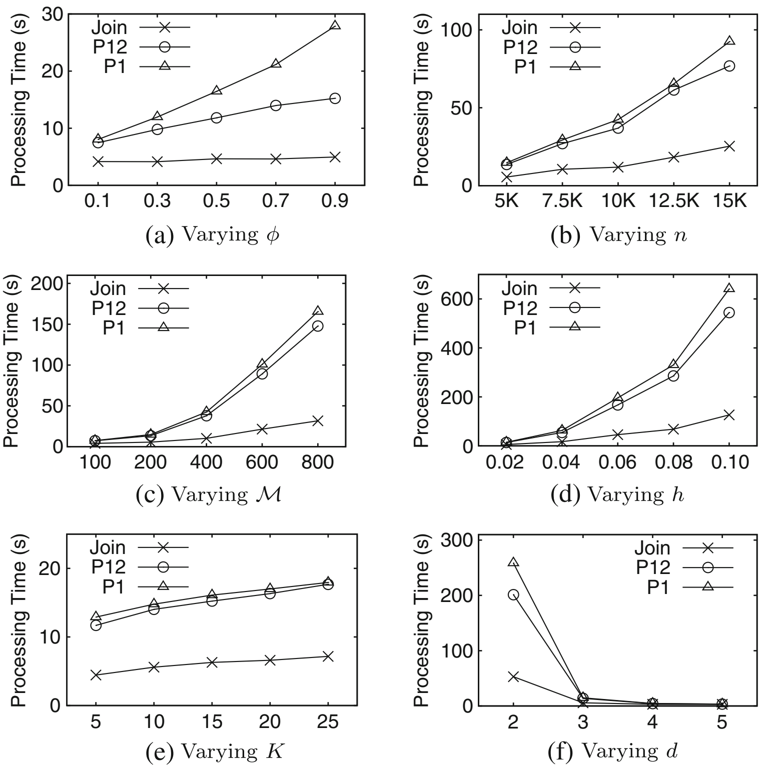


Figure 6 Scalability of **Join**

and hence the pruning power is impaired. With the increase of dimensionality d, when the MBB edge length is fixed, the average area of MBBs gets smaller compared to the whole data space; consequently, the power of the pruning rules becomes more significant. The comparison with **P1** and **P12** illustrates the efficiency of the Pruning Rule 2 and Pruning Rule 3 developed in **Join**. Note that we do not evaluate the performance of **Join** after all three pruning rules are excluded, since Pruning Rule 1 is simple yet very effective. After removing Pruning Rule 1 **Join** algorithm fails to terminate in a reasonable time.

Figure 7 reports the scalability of **G-Join** regarding various parameters compared with **P4** in which only Pruning Rule 4 is applied. As illustrated in the figure, **G-Join** substantially outperforms **P4** in all parameter settings, leading to the conclusion that Pruning Rule 5 is very efficient in practice. The trends observed are similar to those in Figure 6. **G-Join** is not very sensitive to different ϕ and K values. The performance of **G-Join** degrades with larger number of objects and number of instances since more instances are involved in the computations. With a larger average MBB length, **G-Join** requires more computation time since the higher degree of overlapping leads to weaker pruning ability. **G-Join** performs better when the dimensionality increases because once the MBB edge length is fixed, in higher dimensions the average area of an MBB gets smaller so that an object has a larger to be pruned.

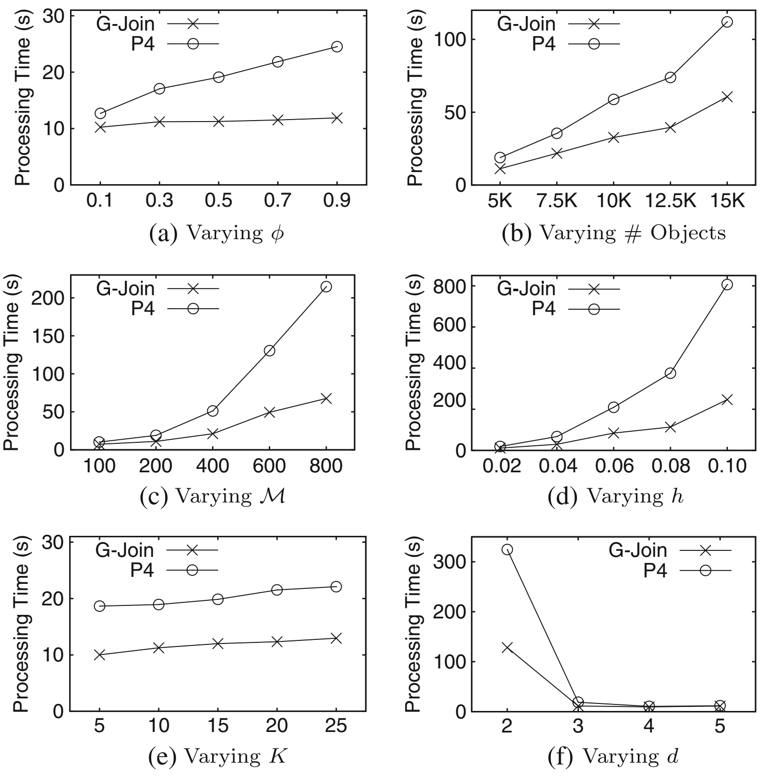


Figure 7 Scalability of **G-Join**

7 Extensions

Techniques proposed in this paper can also be applied to other variations of similarity joins over multi-valued objects. In this section, we define several variations and briefly discuss the techniques.

7.1 Distance threshold based similarity join on multi-valued objects

Top-k and threshold based approaches are two typical mechanism to select a limited number of results which are of most interest to the users. In distance threshold based similarity join on multi-valued objects, a distance threshold γ is pregiven by users according to their preference and domain knowledge, only object pairs with quantile distance not larger than γ will be retrieved.

Problem definition Given a $\phi \in (0, 1]$, a distance threshold γ, two sets of multi-valued objects $\mathcal{U}$ and $\mathcal{V}$ in the d-dimensional space, a threshold based similarity join retrieves pairs of objects from $\mathcal{U} \times \mathcal{V}$ with quantile distance not over γ, namely, $\{(U, V) | U \in \mathcal{U}, V \in \mathcal{V}, dist(U, V) \leq \gamma\}$. Here the distance metrics *dist* could be either ϕ-quantile distance or ϕ-quantile group base distance.

In Algorithm 2, the Seeding phase is no longer required since a distance threshold is pregiven. Similarity join processing algorithms in Sections 4.2 and 5.2 can be directly applied to solving distance threshold based similarity join on multi-valued objects.

7.2 Multi-way distance threshold based similarity join on multi-valued objects

We study pairwise similarity join in this paper where two multi-valued objects sets are combined together to retrieve the most similar object pairs. Multiway spatial joins, on the other hand, involve an arbitrary number of datasets where the join condition is specified over any two datasets. Next we discuss multi-way distance threshold based similarity join on multi-valued objects.

Problem definition Given n multi-valued datasets $\mathcal{U}_1, ..., \mathcal{U}_n$, a distance threshold γ_{ij} is specified between two datasets $\mathcal{U}_i$ and $\mathcal{U}_j$, retrieve all n-objects that respects the distance thresholds, $\{(U_1, ..., U_n) | 1 \leq k \leq n, U_k \in \mathcal{U}_k \ \& \ \forall ij, dist(U_i, U_j) \leq \gamma_{ij}\}$. Here the distance threshold *dist* could be either ϕ-quantile distance or ϕ-quantile group-base distance.

Note that the distance constraint could be applied to any pair of datasets. Mamoulis and Papadias [17] studies multi-way join on spatial data where a *synchronous traversal* paradigm is applied to process the indexes (e.g., R-trees) of all joined datasets. Our filtering techniques could be directly plugged into the framework in [17] to facilitate pruning based on the given pairwise distance constraints.

7.3 Bichromatic and homochromatic top-k similarity join on multi-valued objects

Consider that each object is assigned either blue or red color. A chromatic query adds an additional constraint compared to its nonchromatic version; that is, only the results that meet the color requirement are considered. A bichromatic (homochromatic) top-k similarity join retrieves k most similar object pairs with different (the same) colors.

A straightforward solution for bichromatic and homochromatic top-k similarity join on multi-valued objects based on ϕ-quantile distance or ϕ-quantile group-base distance is to construct two indexes for each color for each dataset involved. Consider the problem definitions in Section 2.1, for the dataset $\mathcal{U}$ we build two R-tree based indexes, $R_{\mathcal{U},\text{red}}$ and $R_{\mathcal{U},\text{blue}}$, for the color red and blue respectively. Similarly, we build $R_{\mathcal{V},\text{red}}$ and $R_{\mathcal{V},blue}$ for dataset $\mathcal{V}$. Bichromatic top-k similarity join will be conducted between the indexes with different colors, i.e., $R_{\mathcal{U},\text{red}} \times R_{\mathcal{V},\text{blue}}$ and $R_{\mathcal{U},\text{blue}} \times R_{\mathcal{V},\text{red}}$, while homochromatic top-k similarity join will be conducted between the indexes with the same color. The pruning techniques developed in our paper could be plugged in to speed up the query processing.

8 Related work

Conventional join queries over two multi-dimensional datasets are fundamental in data analysis and information retrieval. Most existing techniques for join queries have been developed based on popular spatial access methods such as R-trees. For threshold based joins, there are three main stream spatial join algorithms using R*-tree [9]. They are the depth-first-join (DFJ) algorithm [2], the breadth-first-join (BFJ) algorithm [11], and transformation-view-join (TVJ) algorithm [14]. Techniques for top-k spatial/similarity queries are studied in [6, 10]. Various algorithms, such as exhaustive algorithm, recursive algorithm, Heap algorithm, and priority queue based algorithms are proposed. The most recent technique proposes to build an index on the fly [3]. Nevertheless, this technique cannot be used to prune a group of object pairs. That is, every object has to participate in the distance computation. Many variations of join queries over multi-dimensional space have been studied in different contexts, including road networks [22], moving objects [27] and data streams [23].

Spatial queries such as nearest neighbor queries and its variants over fuzzy objects have been studied in [28]. Fuzzy objects possess similar semantics as uncertain objects (e.g., instances are mutually exclusive). The techniques in [28] are not applicable to the problem studied in our paper due to the different semantics as well as inherent difference in query types.

Join queries over uncertain objects are inherently different than conventional joins where each uncertain object takes a set of mutually exclusive instances/points in a multi-dimensional space. It is extensively studied in [4, 13, 16]. Note that all instances in a multi-valued object exist simultaneously instead of mutually exclusive in an uncertain object. Due to such inherent differences in semantics, join techniques over uncertain objects cannot be directly applied to similarity joins over multi-valued objects.

9 Conclusion and future work

We study the problem of top-k similarity join over multi-valued objects. The distance/similarity between two multi-valued objects is measured using quantile based distance to capture the relative instance distribution. A filtering-refinement framework is developed, along with novel, efficient and effective distance, statistic and weight based pruning techniques. Comprehensive experimental study over both real and synthetic datasets demonstrates the efficiency and scalability of our techniques. Possible future directions include identifying the representative ϕ values (i.e., the ϕ values which lead to different query results), and applying other mechanisms such as Borda Count to interpret the semantics of multi-valued objects.

References

1. Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE (2001)
2. Brinkhoff, T., Kriegel, H.P., Seeger, B.: Efficient processing of spatial joins using r-trees. In: SIGMOD (1993)
3. Cheema, M.A., Lin, X., Wang, H., Wang, J., Zhang, W.: A unified approach for computing top-k pairs in multidimensional space. In: ICDE (2011)
4. Cheng, R., Singh, S., Prabhakar, S., Shah, R., Vitter, J.S., Xia, Y.: Efficient join processing over uncertain data. In: CIKM (2006)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms, 2nd edn., chapter 9: medians and order statistics. In: The MIT Press (2009)
6. Corral, A., Manolopoulos, Y., Theodoridis, Y., Vassilakopoulos, M.: Closest pair queries in spatial databases. In: SIGMOD (2000)
7. Elmasri, R., Navathe, S.: Fundamentals of database systems, 6th edn. (2011)
8. Guntzer, M.M., Jungnickel, D.: Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem. In: Operations Research Letters (2000)
9. Han, W.S., Kim, J., Lee, B.S., Tao, Y., Rantzau, R., Markl, V.: Cost-based predictive spatiotemporal join. In: TKDE (2009)
10. Hjaltason, G., Samet, H.: Incremental distance join algorithms for spatial databases. In: SIGMOD (1998)
11. Huang, Y.W., Ning, J., Rundensteiner, E.A.: Spatial joins using r-trees: breadth-first traversal with global optimizations. In: VLDB (1997)
12. Knorr, E.M., Ng, R.T.: Finding aggregate proximity relationships and commonalities in spatial data mining. In: TKDE (1996)
13. Kriegel, H.P., Kunath, P., Pfeifle, M., Renz, M.: Probabilistic similarity search on uncertain data. In: DASFAA (2006)
14. Lee, M.J., Whang, K.Y., Han, W.S., I.-Y, S.: Transform-space view: performing spatial join in the transform space using original-space indexes. In: TKDE (2006)
15. Lin, X., Zhang, Y., Zhang, W., Cheema, M.A.: Stochastic skyline operator. In: ICDE (2011)
16. Ljosa, V., Singh, A.K.: Top-k spatial join of probabilistic objects. In: ICDE (2008)
17. Mamoulis, N., Papadias, D.: Multiway spatial joins. In: TODS (2001)
18. Meester, R.: A natural introduction to probability theory. Springer (2008)
19. Musial, K., Budka, M., Juszczyszyn, K.: Creation and growth of online social network. In: World Wide Web Journal (2012)
20. Papadias, D., Kalnis, P., Zhang, J., Tao, Y.: Efficient OLAP operations in spatial data warehouses. In: SSTD (2001)
21. Rigaux, P., Scholl, M., Voisard, A.: Spatial databases: with applications to gis. Morgan Kaufmann (2002)
22. Sankaranarayanan, J., Alborzi, H., Samet, H.: Distance join queries on spatial networks. In: GIS (2006)
23. Shen, Z., Cheema, M.A., Lin, X., Zhang, W., Wang, H.: Efficiently monitoring top-k pairs over sliding windows. In: ICDE, pp. 798–809 (2012)
24. Wei, F., Qian, W., Wang, C., Zhou, A.: Detecting overlapping community structures in networks. In: World Wide Web Journal (2009)
25. Yiu, M.L., Mamoulis, N., Tao, Y.: Efficient quantile retrieval on multi-dimensional data. In: EDBT (2006)

26. Zhang, W., Lin, X., Cheema, M.A., Zhang, Y., Wang, W.: Quantile-based knn over multi-valued objects. In: ICDE (2010)
27. Zhang, R., Lin, D., Ramamohanarao, K., Bertino, E.: Continuous intersection joins over moving objects. In: ICDE (2008)
28. Zheng, K., Fung, P., Zhou, X.: K nearest neighbor search for fuzzy objects. In: SIGMOD (2010)