Consensus-based Ranking of Multi-valued Objects: A Generalized Borda Count Approach

Ying Zhang[1] Wenjie Zhang[1] Jian Pei[2] Xuemin Lin[1] Qianlu Lin[1] Aiping Li[3]

[1] The University of New South Wales , {yingz,zhangw,lxue, qlin}@cse.unsw.edu.au

[2] Simon Fraser University, jpei@cs.sfu.ca

[3] National University of Defense Technology, apli1974@gmail.com

Abstract—In this paper, we tackle a novel problem of ranking multi-valued objects, where an object has multiple instances in a multidimensional space, and the number of instances per object is not fixed. Given an ad hoc scoring function that assigns a score to a multidimensional instance, we want to rank a set of multi-valued objects. Different from the existing models of ranking uncertain and probabilistic data, which model an object as a random variable and the instances of an object are assumed exclusive, we have to capture the coexistence of instances here. To tackle the problem, we advocate the semantics of favoring widely preferred objects instead of majority votes, which is widely used in many elections and competitions. Technically, we borrow the idea from Borda Count, a well recognized method in consensus-based voting systems. However, Borda Count cannot handle multi-valued objects of inconsistent cardinality, and is costly to evaluate top-k queries on large multidimensional data sets. To address the challenges, we extend and generalize Borda Count to quantile-based Borda Count, and develop efficient computational methods with comprehensive cost analysis. We present case studies on real data sets to demonstrate the effectiveness of the generalized Borda Count ranking, and use synthetic and real data sets to verify the efficiency of our computational method.

Index Terms—Multi-valued Objects, Consensus-based Ranking

1 Introduction

In several applications, such as analyzing economic data, data is often modeled as multi-valued objects. For example, to compare the household income among several cities, we often randomly collect the household income of a set of individuals as samples from each city. Then, cities are compared using the sample sets. In this case, each city is represented as a multi-valued object, each value, also known as an instance, being a sample. Another example is the evaluation of the research groups where each research group is a multi-valued object, and the teaching and researching performance evaluations of each staff correspond to one instance. Due to various factors, such as the different availability of samples in different cities, the number of samples per city is not fixed. Similarly, the sizes of two research groups may be different. According to the significance of the instances, e.g., the size of a family and the position of a staff, the instances may carry different weights.

As ranking is an essential analytic method, it is natural and fundamental to investigate how to rank a set of multi-valued objects. To the best of our knowledge, however, there is no existing work addressing this important problem systematically. One may think we may simply rank multi-valued objects as uncertain/probabilistic data. However, the models of multi-valued objects and uncertain/probabilistic ones are fundamentally different. All instances of an uncertain/probabilistic object are assumed exclusive – only up to one instance can appear at a time [23]. Therefore, the uncertain objects can be ranked based on the *possible world semantics*. In contrast, all instances of a multi-valued object are assumed coexisting, and hence we cannot rank the multi-valued objects by uncertain object models. Appendix C provides the detailed introduction for the uncertain object model and discussion of the difference between the uncertain object model and the multi-valued object model. Moreover, the existing methods, such as expected value, median and quantile, summarize the multiple instances of an object using an aggregate function that tries to capture the central tendency (i.e., the majority) of an object, and then rank objects according to the aggregates. We argue that majority-based ranking may not be appropriate for multi-valued objects. Instead, we propose a consensus-based ranking approach, which prefers objects widely ranked high, serves the task of ranking multi-valued objects better.

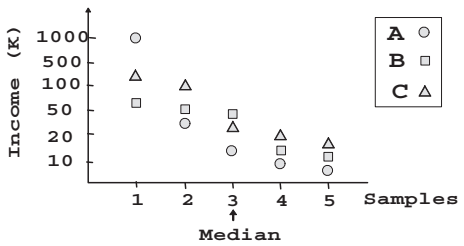


Fig. 1. A synthesized data set in Example 1

Example 1 (Motivation)**.** *Suppose we want to compare the annual household income of 3 cities, namely A, B and C. Each city is modeled as a multi-valued object*

and has 5 instances (with the same weight) whose incomes are shown in Fig 1 where instances of each object are sorted in descending order based on their incomes. Particularly, we have A=$\{1000K, 30K, 15K, 10K, 5K\}$, B=$\{400K, 100K, 40K, 20K, 15K\}$, *and* C=$\{60K$,$52K$,$50K$, $15K$, $12K\}$. *To rank the 3 cities, one may calculate the average income for each city, and then sort the cities on average income. It is well known that the average income based ranking is sensitive to the outliers. In the example, A is ranked the first because its rank is boosted by the first sample which has much higher income, although the remaining of the samples have very low incomes. Another alternative is to rank cities based on their ϕ-quantile income where $0 < \phi \leq 1$. The quantile based approach is widely employed in various literatures to summarize the distributions [18]. However, a single quantile (e.g., median) cannot capture the score distribution. Moreover, it is difficult for users to set a proper ϕ value and the rank of a city may vary dramatically regarding different ϕ values. In the example, B is ranked the 1st, the 2nd and the 3rd positions when ϕ equals 0.5 (median), 0.3 and 0.1 respectively.*

Observe that a single ϕ value cannot capture the score distributions of the multi-valued objects, in the paper we aim to develop a new consensus-based ranking approach such that the ranking result regarding different ϕ values are carefully considered in a comprehensive way. For each given $\phi \in (0, 1]$, there is a ranking list according to the ϕ-quantile values of the objects. Therefore, it is intuitive to apply the rank aggregation approaches [11], [22] (See Section 7.3) to compute the rank of the multi-valued objects. The main challenge lies in the fact that the number of possible ϕ value is infinite, and hence we cannot directly apply the existing approaches. Due to its simplicity and popularity, we adopt the Borda Count (BC for short) method [10] for the rank aggregation.

In this paper, we will develop models and algorithms for consensus-based ranking of multi-valued objects. Our idea of consensus-based ranking is an extension of consensus-based voting. Specifically, we start from Borda Count, a typical consensus-based voting method. BC and its variations have been popularly adopted by many private organizations and competitions all-over the world since the 18th century, such as determining awards for sports in the US (e.g., the NBA MVP selection, ranking players in NCAA), achieving consensus for non-electoral purpose in Northern Ireland, electing student governments and officers in a few US universities and professional societies, and selecting features in OpenGL Architecture Review Board.

In the BC method [10], suppose there are n voters and m candidates. Every voter ranks the m candidates according to her/his preference. Denote by $r_i = j$ $(0 \leq j < m, 1 \leq i \leq n)$ that the i-th voter ranks a candidate at at the $(j+1)$-th position. A candidate has an average rank $\sum_{i=1}^{n} \frac{r_i}{n}$. The m candidates are ranked by their average ranks.

Example 2. *Regarding the example in Fig. 1, we assume there are 5 voters and the i-th voter ranks three objects regarding their i-th samples. Recall that the samples of each multi-valued object are sorted in descending order of their incomes. Particularly, the object A is ranked at the 1st, the 3rd, the 3rd, the 3rd and the 3rd positions by the 1st, 2nd, 3rd, 4th and 5th voters respectively. Consequently, the average rank of A is $\frac{8}{5}$. Similarly, the average rank of B and C is $\frac{5}{5}$ and $\frac{2}{5}$ respectively. It is quite intuitive that C has the highest rank because C is widely preferred. Ranking in this way is based on the general agreement among the rankings of the 5 positions instead of central tendency of the income data.*

BC by itself is a single-winner approach. To meet the need of data analytics on large data sets, we propose to follow the popularly adopted theme of retrieving the top-k objects. We address several challenges.

First, a straightforward extension of the BC method can only handle the cases where every object has the same number of instances. However, in many applications, different objects may have different cardinalities. For example, in an economic data analysis, different cities may likely have different numbers of instances. To tackle the problem, we develop a generalized BC ranking model based on quantiles. The central idea is to consider all possible rankings at different quantiles in deriving the overall ranking.

Second, more often than not, a multi-valued object as well as all instances of the object are in a multidimensional space. For example, an economic data set may contain the information about household income, housing cost, and family size, which is in a 3-dimensional space. A user may apply an ad hoc preference function to calculate a score for each instance of an object, and then uses those values to rank the objects. Then, how to rank multi-dimensional, multi-valued objects is far from trivial. To tackle the problem, we develop several efficient techniques to compute the top-k objects.

Our principle contributions in this paper can be summarized as follows.

- A novel consensus-based ranking method, named BC ranking, is proposed for the problem of top k query on *multi-valued* objects.
- Effective and efficient algorithms are developed to compute the top k query based on BC ranks. Effective pruning techniques are proposed to significantly improve the performance in terms of CPU and I/O costs.
- A cost model is proposed to analyze the I/O costs of the algorithms. Experiments demonstrate that our cost model is highly accurate.
- We present case studies on real data sets to demonstrate the effectiveness of the generalized BC ranking. We also use synthetic and real data sets to verify the efficiency of our computational method.

The rest of the paper is organized as follows. Section 2 formalizes the problem. Section 3 discusses how to reduce the search space. Section 4 presents an efficient index based top-k query answering algorithm. Section 5 discusses the extension of the techniques and the future

Notation	Meaning
U, V	Multi-valued objects
$\mathcal{U}$, $\mathcal{S}$	set of *multi-valued* objects
f	preference function
U_f	score distribution of U regarding f
u, v	instances of the *multi-valued* objects
n	the number of *multi-valued* objects
m	the average number of instances per object
U_{mbb}	minimal bounding box of U
$f^-(U)(f^+(U))$	minimal(maximal) score regarding U_{mbb} and f
$f_\phi(U)$	ϕ-quantile score of U regarding f
$\phi_{\gamma,f}(U)$	accumulated instance weights of U regarding score γ and f
$\bar{\phi}_{\gamma,f}(U)$	an upper bound function of $\phi_{\gamma,f}(U)$ regarding score γ and f
$r_{\phi,f}(U)$	ϕ-quantile rank of U regarding f
$r(U)$	Borda Count(BC) based rank of U
$\phi(u)$	*distinguished* quantile value of u
$r^-(U)$	lower bound of $r(U)$
U_{stat}	indicate whether U is an *active* object

TABLE 1
The summary of notations.

work. Section 6 evaluates the effectiveness, efficiency and scalability of our techniques. Section 7 introduces the related work and Section 8 concludes the paper. Due to space limitation, the following Sections are presented in the **Appendices of the paper in the supplemental file**. Specifically, Appendices are organized as follows. Detailed proof for the Theorem 3 is presented in Section A. Section B proposes a cost model and analyzes the I/O cost for algorithms proposed in the paper. In Section C, we introduce the uncertain object model and quantile rank based approach for uncertain object, as well as the discussion on the difference between the uncertain object model and multi-valued object model. This is followed by some experiments in Section D.

Table 1 summarizes the notations frequently used in the paper.

2 Background

In this paper, a point p in a d-dimensional numerical space R^d is used to represent a record with d features. The coordinate value of p on the i-th dimension, denoted by $p.D_i$, corresponds to the i-th feature value of the record. A top-k query consists of a preference function f and a result size k, where $f(p)$ defines a numerical score for each point $p \in R^d$. Without loss of generality, we assume that *smaller scores are preferred*. For simplicity in presentation, we focus on *linear additive functions* in this paper, i.e., $f(p) = \sum_{i=1}^{d} a_i p.D_i$, where a_i ($a_i > 0$) is a weight ($1 \leq i \leq d$). Without loss of generality, we assume $\sum_{i=1}^{d} a_i = 1$. Note that linear additive functions are among the most popular preference functions [8]. Section 5 demonstrates that our techniques can be naturally extended to other preference functions.

A multi-valued object is represented as $U = \{(u_i, w(u_i)) | 1 \leq i \leq m\}$ where u_i is an instance (point),

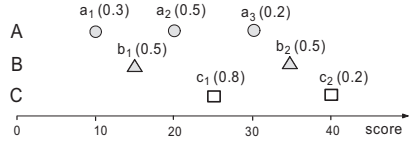


Fig. 2. Score Distributions

$0 < w(u_i) \leq 1$ and $\sum_{i=1}^{m} w(u_i) = 1$. The weight of an instance reflects its significance for the *multi-valued* object. For simplicity in presentation, a *multi-valued* object is abbreviated as an object whenever there is no ambiguity.

Given an object U and a preference function f, the score of U regarding f corresponds to a **score distribution** $U_f = \{f(u), w(u)\}$ for all instances $u \in U$, where $f(u)$ denotes the score of u regarding f, and $w(u)$ is the weight of u. For simplicity in presentation, in this paper we assume the instances of an object are ordered by their scores regarding f; that is, for any two instances u_i and u_j of an object U, we have $f(u_i) \leq f(u_j)$ if $i < j$.

Example 3. *In Fig. 2, the object A has three instances* a_1, a_2 *and* a_3 *with scores* $f(a_1) = 10$, $f(a_2) = 20$ *and* $f(a_3) = 30$ *respectively and weights* 0.3, 0.5 *and* 0.2 *respectively. We have* $A_f = \{(10, 0.3), (20, 0.5), (30, 0.2)\}$. *Similarly,* $B_f = \{(15, 0.5), (35, 0.5)\}$, *and* $C_f = \{(25, 0.8), (40, 0.2)\}$.

We extend BC to rank a set of objects as follows.

Definition 1 (BC method)**.** *Consider* n_v *voters. Let* $\pi_i(U)$ *denote the position of the object* U *in the* i*-th voter's rank list. Note that in this paper* $\pi_i(U) = 0$ *if an object* U *takes the first position. For each object* U*, a score* $\sum_{i=1}^{n_v} \pi_i(U)/n_v$ *is assigned. Then, the* k *objects with the smallest scores are returned as the top* k *result.*

To rank multi-valued objects, we need to use quantiles.

Definition 2 (ϕ-quantile score)**.** *Given* $\phi \in (0, 1]$*, the* ϕ*-quantile score of an object* U *regarding* f*, denoted by* $f_\phi(U)$*, is the* ***score*** *of the first instance* u_i *such that* $\sum_{j=1}^{i} w(u_j) \geq \phi$.

Clearly, f_ϕ is a non-decreasing function.

Example 4. *Based on Example 3, Fig. 3 depicts the* ϕ*-quantile scores of an object as a function of the* ϕ *values. We have* $f_\phi(A) = 20$ *when* $0.3 < \phi \leq 0.8$ *according to Definition 2. Similarly,* $f_\phi(B) = 15$ *when* $\phi \in (0, 0.5]$.

To rank multi-valued objects where the number of instances per object is not fixed, each ϕ-quantile for $\phi \in (0, 1]$ is regarded as a voter, and ranks objects according to their corresponding ϕ-quantile scores. Then, the *Borda Count based rank* (BC rank for short) of each object can be calculated according to the BC method. This is a generalization of the BC rank method.

Definition 3 (ϕ-quantile rank)**.** *Given a set* $\mathcal{U}$ *of objects and a preference function* f*, the* ϕ*-quantile rank of an object* U *regarding* f*, denoted by* $r_{\phi,f}(U)$*, is*

$$r_{\phi,f}(U) = |\{V | f_\phi(V) < f_\phi(U),\ V \in \mathcal{U} - U\}| \qquad (1)$$

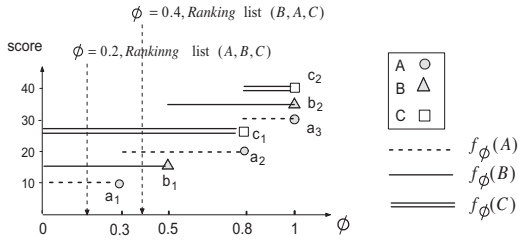


Fig. 3. ϕ quantile score example

Example 5. *In Fig. 3, the rank list of objects is* (A, B, C) *when* $\phi = 0.2$, *and* (B, A, C) *when* $\phi = 0.4$. *According to Definition 3,* $r_{0.2,f}(A) = 0$, $r_{0.2,f}(B) = 1$, $r_{0.2,f}(C) = 2$, $r_{0.4,f}(A) = 1$, $r_{0.4,f}(B) = 0$, *and* $r_{0.4,f}(C) = 2$.

As all possible quantile values within $(0, 1]$ are considered as voters, the number of voters is infinite when BC ranking is applied. Suppose quantile values are divided into n_v groups by ϕ_0, ϕ_1, ϕ_2, $\ldots$, ϕ_{n_v} where $\phi_i = i \times \Delta_\phi$ and $\Delta_\phi = \frac{1}{n_v}$. When $\Delta_\phi \to 0$, we have $r_{\phi,f}(U) \approx r_{\phi_i,f}(U)$ for any $\phi \in (\phi_{i-1}, \phi_i]$. In the light of BC method, the BC rank of an object U, denoted by $r(U)$, is defined as follows.

Definition 4 (BC based rank)**.**

$$r(U) = \lim_{\Delta_\phi \to 0} \sum_{i=1}^{n_v} r_{\phi_i,f}(U)\Delta_\phi = \int_0^1 r_{\phi,f}(U)d(\phi) \qquad (2)$$

Note that an object with a smaller BC rank is preferred (i.e., ranked higher) in a top-k query.

Problem Statement. Given a set $\mathcal{U}$ of n *multi-valued* objects and a preference function f, we investigate the problem of retrieving the top k objects with the highest BC based ranks. Without loss of generality, we assume $n > k$ and ties are broken arbitrarily.

In Equation 2, for an object U, since $r_{\phi,f}(U)$ is not pre-computed and the number of possible ϕ values is infinite, it is infeasible to directly compute $r(U)$. Section 3 shows how to reduce to a finite search space.

3 Search Space Reduction

In this section, we show that only a limited number of ϕ values are required for the computation of BC ranks. For each instance $u_i \in U$, let $\phi(u_i) = \sum_{j=1}^{i} w(u_j)$, which is called a *distinguished* quantile of U. Let $\Phi_f(U) = \{\phi(u) | u \in U\}$ represent a set of *distinguished* quantiles of U regarding f. For presentation simplicity, we assume every object has a dummy instance u_0 where $\phi(u_0) = 0$. According to Definition 2, given an object U with m instances, we have $f_\phi(U) = f_{\phi_i}(U)$ if $\phi(u_{i-1}) < \phi \leq \phi(u_i)$ where $1 \leq i \leq m$.

Example 6. *In Fig. 3, we have* $\Phi_f(A) = \{0.3, 0.8, 1\}$. *Similarly,* $\Phi_f(B) = \{0.5, 1\}$ *and* $\Phi_f(C) = \{0.8, 1\}$. *According to Definition 2, we have* $f_\phi(A) = 10$ *for* $\phi \in (0, 0.3]$, $f_\phi(A) = 20$ *for* $\phi \in (0.3, 0.8]$ *and* $f_\phi(A) = 30$ *for* $\phi \in (0.8, 1]$.

Algorithm 1: *Quantile Based Algorithm*($\mathcal{U}$, f, k)

Input : $\mathcal{U}$: a set of objects to be ranked,
f : preference function, k
Output : top k objects regarding f
1 Compute and sort instances of U for all $U \in \mathcal{U}$ regarding f;
2 $\mathcal{Q} := \bigcup \Phi_f(U)$ for all $U \in \mathcal{U}$;
3 sort $\mathcal{Q}$ in increasing order;
4 $T := 0$; $r(U) := 0$ for all $U \in \mathcal{U}$;
5 **for** each ϕ in $\mathcal{Q}$ accessed in order **do**
6 Sort objects in $\mathcal{U}$ based on their ϕ-quantile scores ;
7 **for** each $U \in \mathcal{U}$ **do**
8 $r(U) := r(U) + r_{\phi,f}(U) \times (\phi - T)$;
9 $T := \phi$;
10 **return** k objects with highest BC ranks

Nevertheless, the ϕ-quantile rank ($r_{\phi,f}(U)$) of U may change between two consecutive *distinguished* quantiles in $\Phi_f(U)$. This is because the ϕ-quantile scores of other objects change and lead to different ϕ-quantile ranks of U.

Let $\mathcal{Q}$ denote the union of all *distinguished* quantiles of all objects, i.e., $\mathcal{Q} = \bigcup \Phi_f(U)$ for $U \in \mathcal{U}$, Theorem below indicates the ϕ-quantile rank of an object remains unchanged for any two consecutive *distinguished* quantiles in $\mathcal{Q}$.

Theorem 1. *Let* $\mathcal{Q} = \bigcup \Phi_f(U)$ *for all objects* $U \in \mathcal{U}$. *Suppose the distinguished quantiles in* $\mathcal{Q}$ *are in increasing order, then we have*

$$r(U) = \sum_{i=1}^{|\mathcal{Q}|} (\phi_i - \phi_{i-1}) \times r_{\phi_i,f}(U) \qquad (3)$$

The correctness of Theorem 1 is immediate since the ϕ-quantile scores of all objects remain unchanged for any two consecutive *distinguished* quantiles in $\mathcal{Q}$.

Example 7. *In Fig. 3, we have* $\mathcal{Q} = \{0.3, 0.5, 0.8, 1\}$. *We have* $r_{\phi,f}(A)$ *equals* 0, 1, 0 *and* 0 *when* ϕ *falls in* $(0, 0.3]$, $(0.3, 0.5]$, $(0.5, 0.8]$ *and* $(0.8, 1]$ *respectively. Therefore,* $r(A) = 0 \times 0.3 + 1 \times 0.2 + 0 \times 0.3 + 0 \times 0.2 = 0.2$. *Similarly, we have* $r(B) = 1.1$ *and* $r(C) = 1.7$. *The top* 2 *result is* $\{A, B\}$.

Algorithm 1 illustrates an implementation of BC rank based top k algorithm according to Theorem 1. For each $\phi \in \mathcal{Q}$, Lines 6-9 sort the objects in $\mathcal{U}$ by their ϕ-quantile scores and update their BC ranks. In Line 9, T keeps the largest *distinguished* quantile processed so far. Line 10 returns k objects with highest BC ranks. Recall that V has higher BC rank than U if $r(V) < r(U)$.

Example 8. *In Fig. 3, when* $\phi = 0.5$, *we have* $T = 0.3$, $r(A) = 0$ *and* $r_{\phi,f}(A) = 1$. *Therefore,* $r(A) = 0 + 1 \times (0.5 - 0.3) = 0.2$ *in Line 8.*

Let $|\mathcal{Q}|$ denote the number of distinct *distinguished* quantiles in $\mathcal{Q}$, the time complexity of Algorithm 1 is $O(nm\log(m) + |\mathcal{Q}| \times n\log(n))$=$O(nm\log(m)$

$+n^2 m \log(n))$, where n and m are the number of objects and the average number of instances per object, respectively.

4 An Index-Based Algorithm

In Algorithm 1, the ranks of the objects are computed *simultaneously*, which leads to high CPU and I/O costs. Users are often interested in only the top k objects with the highest BC ranks. This motivates us to develop an *index based* algorithm to *sequentially* compute the ranks of the objects such that many unpromising objects can be eliminated by filtering and pruning techniques, and hence reduce the CPU and I/O costs.

According to the definition of ϕ-quantile rank (Equation 1), we rewrite Equation 2 as follows.

$$r(U) = \sum_{V \in \mathcal{U} \setminus U} \Delta_{U,V} \quad (4)$$

where $\Delta_{U,V}$ denotes the contribution of V to $r(U)$; that is, $\Delta_{U,V} = \int_0^1 \xi(\phi) d(\phi)$ where $\xi(\phi) = 1$ if $f_\phi(V) < f_\phi(U)$ otherwise $\xi(\phi) = 0$. This implies that we can calculate $r(U)$ based on $\Delta_{U,V}$ for each $V \in \mathcal{U} \setminus U$. Note that if $U = V$, we have $\Delta_{U,V} = 0$ and $\Delta_{V,U} = 0$.

Suppose the instances of two objects U and V are ordered by their scores regarding f respectively, we can compute $\Delta_{U,V}$ and $\Delta_{V,U}$ **at the same time** with $O(m)$ time following the similar rationale of Algorithm 1, where $\mathcal{Q} = \Phi_f(U) \bigcup \Phi_f(V)$. Recall that m is the average number of instances per object.

Example 9. *In Fig. 3, we have $\Delta_{A,B} = 0.2$ and $\Delta_{B,A} = 0.8$ because $f_\phi(A) > f_\phi(B)$ if $\phi \in (0.3, 0.5]$ and otherwise $f_\phi(A) < f_\phi(B)$. Similarly, we have $\Delta_{A,C} = 0$ and $\Delta_{C,A} = 1$. Therefore, $r(A) = \Delta_{A,B} + \Delta_{A,C} = 0.2$ according to Equation 4.*

The time complexity of the algorithm is $O(nm \log(m) + n^2 m)$ in the worse case because we may have to calculate $\Delta_{A,B}$ for every pair of objects A and B in $\mathcal{U}$. Therefore, the key of the algorithm is to develop effective and efficient filtering and pruning techniques such that a large number of non-promising objects can be eliminated, and hence save CPU and I/O costs.

In this section, we propose a framework of the index based algorithm in Section 4.1, which consists of *filtering* and *refinement* phases. Section 4.2 introduces an R-Tree based filtering technique to eliminate the non-promising objects. Then the refinement algorithm is presented in Section 4.3.

4.1 Framework

Given an object U, let U_{mbb} denote the Minimal Bounding Box (MBB) of the instances of U; that is, the U_{mbb} is the **minimal** bounding box such that for any instance $u \in U$, u is located within U_{mbb}. The computation of the MBB of the object is cheap, i.e., only need one scan of the instances, and it is space efficient to maintain MBBs of the objects. Therefore, in this paper, we assume the MBBs of all objects are available and are organized by an R-Tree [14], denoted by $R_\mathcal{U}$. Particularly, the MBB of an object corresponds to a data entry of $R_\mathcal{U}$. Note that our techniques can be easily adapted to other spatial index techniques, since the algorithm follows the standard *branch and bound* paradigm [6].

Algorithm 2: ***Index Based Algorithm***($R_\mathcal{U}$, f, k)

Input : $R_\mathcal{U}$: the R-Tree for a set of objects $\mathcal{U}$, f : the preference function, k
Output : top k objects regarding BC ranks

1 $\mathcal{S} \leftarrow$ **Filtering**(R_U, f, k);
2 **return** **Refinement**(S, f, k)

Algorithm 2 presents the framework of our index based algorithm. Line 1 applies the filtering technique based on R_U, i.e., the R-Tree based on MBBs of the objects, to eliminate some non-promising objects. Line 2 refines the remaining objects and return the top k objects with highest BC ranks. The details of *Filtering* and *Refinement* algorithms will be presented in Section 4.2 and Section 4.3 respectively.

4.2 Filtering

Given an object U and a linear preference function f, $f^-(U)$ ($f^+(U)$) denotes the minimal (maximal) score of U regarding U_{mbb} and f. Because of the linearity of the preference function, it is immediate that we have $f^-(U) \leq f(u) \leq f^+(U)$ for any $u \in U_{mbb}$.

Intuitively, for two objects A and B, A should be ranked higher than B if $f^+(A) < f^-(B)$ because the worst instance (with the maximal score value) of A can still outperform the best instance (with the minimal score value) of B regardless of the quantile ϕ values. Recall that we assume the lower score values are preferred.

This motivates us to develop efficient *filtering* technique to reduce the top k candidate size based on the MBBs of the objects (i.e., $f^-(U)$ and $f^+(U)$ for any object U), and hence significantly reduce the CPU and I/O costs. Theorem below indicates that we can safely exclude some objects from the top k computation based on the minimal and maximal scores derived from their MBBs and the linear preference function f.

Theorem 2. *Let f_k^+ be the k-th smallest f^+ score of the objects in $\mathcal{U}$, then only object U with $f^-(U) \leq f_k^+$ can be top k candidates regarding f. Let f_s^+ be the largest f^+ score among these candidate objects, any object V with $f^-(V) > f_s^+$ can be excluded from the top k computation.*

Proof: Given two objects U and V with $f^+(V) < f^-(U)$, it is immediate that $r_{\phi,f}(V) < r_{\phi,f}(U)$ for any $\phi \in (0, 1]$, and hence $r(V) < r(U)$. An object U can be excluded from top k candidates if there exist k other objects $\{V\}$ such that $f^+(V) < f^-(U)$. Therefore, only object U with $f^-(U) \leq f_k^+$ can be top k candidates regarding f. Let f_s^+ be the largest f^+ score among the top k candidate objects, any object V with $f^-(V) > f_s^+$

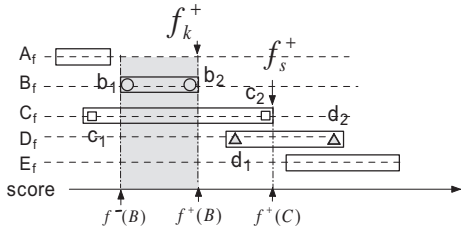


Fig. 4. top k example for $k = 2$

can be excluded from computation because V does not contribute to the BC rank of any top k candidate object, i.e., $\Delta_{U,V} = 0$ for any candidate object U. □

Example 10. *In Fig. 4, we have $f_k^+ = f^+(B)$ when $k = 2$ and hence the top k candidates $\mathcal{C} = \{A, B, C\}$. Consequently, we have $f_s^+ = f^+(C)$ and E can be pruned since $f^-(E) > f_s^+$. Note that although D is not a top k candidate, it contributes to the computation of $r(C)$. Therefore, we have $\mathcal{S} = \{A, B, C, D\}$, where $\mathcal{S}$ denotes the set of objects survived from filtering procedure.*

For an object U, we use U_{stat} to indicate if U is an *active* object (i.e., top k candidate). Clearly, we only need to compute BC rank of the objects $\{U\}$, where U_{stat} is set to *active*. Note that besides the top k candidates, we also need to keep objects which may contribute to the rank of the *active* objects. For instance, in Fig. 4 suppose $w(b_1) = w(b_2) = 0.5$, $w(c_1) = 0.6$, $w(c_2) = 0.4$, $w(d_1) = 0.9$ and $w(d_2) = 0.1$, we have $r(C) < r(B)$ if D is not considered, otherwise, it turns out $r(B) < r(C)$.

An Efficient Filtering Algorithm. We can first compute the $f^-(U)$ and $f^+(U)$ for each object $U \in \mathcal{U}$, and then apply Theorem 2 by sorting objects based on their f^+ values. However, all MBBs of the objects will be accessed which is less efficient. Below, we show how to efficiently apply the filtering based on $R_\mathcal{U}$.

As illustrated in Algorithm 3, we eliminate the objects based on Theorem 2. A min heap H is employed to keep the entries of $R_\mathcal{U}$, where the key of an entry e is its minimal score regarding f, denoted by $f^-(e_{mbb})$. The algorithm consists of two phases. In the first phase (phase *one*), we use f_k^+ to maintain the k-th smallest f^+ values for the objects seen so far. An object U visited during phase *one* is a top k candidate, and U_{stat} is set to *active* in Line 12. Moreover, Line 11 updates f_k^+ based on max ($f^+(U)$). Recall that max records the score upper bound (i.e., $f^+(U)$) of the most recently visited object U. Specifically, a priority queue with maximal size k is employed to maintain the k objects with the smallest f^+ values (i.e., f_k^+) seen so far. Meanwhile, we keep the largest f^+ value among the objects visited for f_s^+. Recall that f_s^+ is the largest f^+ value among the candidate objects, i.e., the objects visited in the phase *one*. Algorithm 3 goes to the second phase (phase *two*) in Line 9 once the minimal score of the current object is larger than f_k^+, which implies that all of the unvisited

Algorithm 3: ***Filtering***($R_\mathcal{U}$, f, k)

```
   Input  : R_U : the R-Tree for a set of objects U,
            f : the preference function, k
   Output : S : objects survived from filtering
1  f_k^+ := ∞; f_s^+ := 0; S := ∅; H := ∅ ;
2  insert root of R_U into the heap H;
3  while H ≠ ∅ do
4      e := Deheap(H);
5      min := f^-(e_mbb); max := f^+(e_mbb);
6      if e corresponds to the object U then
7          if In phase one then
8              if min > f_k^+ then
9                  algorithm status becomes phase two;
10             else
11                 Update f_k^+ and f_s^+ based on max;
12                 U_stat := active;
13         S := S ∪ U if in phase one or min < f_s^+ ;
14     else
15         if In phase one or min < f_s^+ then
16             load child entries of e ;
17             for each child entry e_i do
18                 if In phase one or f^-(e_{i_mbb}) < f_s^+ then
19                     put e_i into heap H;
20 return S
```

objects (including the current one) are not top k candidate objects. Nevertheless, we also keep an object U with $f^-(U) \leq f_s^+$ because it may contribute to the rank computation of the candidate objects. Lines 15-19 put child entries of e into heap H for further computation if the algorithm is in the first phase or the minimal score of e_{mbb} regarding f is smaller than f_s^+. In our top k algorithm, we only need to consider the objects in $\mathcal{S}$, i.e., objects survived from the **filtering** phase.

Correctness. In Algorithm 3, objects are accessed in non-decreasing order regarding their f^- values. As the value of f_k^+ for objects seen as far is non-increasing, we have $f_k^+ < f^-(U) \leq f^+(U)$ for any object U accessed after phase *one*. Therefore, f_k^+ and f_s^+ are correctly calculated at the end of phase *one*. In the *second* phase, an intermediate entry e is not processed if $f^-(e_{mbb}) > f_s^+$ since $f^-(e_{i_{mbb}}) > f_s^+$ for any child entry e_i of e. Therefore, the correctness of Algorithm 3 holds.

Time Complexity. In Lines 4 and 19 of Algorithm 3, it takes time $O(\log(n))$ to maintain the heap H, where n is the number of the objects in $\mathcal{U}$. The update of f_k^+ (Line 11) costs time $O(\log(k))$ by using a priority queue. Since n is usually much larger than k, the time cost of Algorithm 3 is $O(n_e \log(n))$, where n_e denotes the number of entries (data entries and intermediate entries) visited in Algorithm 3. A comprehensive analysis on $|S|$ and the I/O cost of Algorithm 3 is presented in Appendix B.

4.3 Refinement

According to Theorem 2, we only need to compute the BC ranks of the *active* objects in $\mathcal{S}$ based on Equation 4

to find the top k objects. Nevertheless, the costs may still be expensive due to the possiblely large number of *active* objects and the I/O cost incurred when the instances of the objects are loaded.

In this subsection, we first introduce two important pruning rules, namely *dominance based* pruning and *rank based* pruning respectively, to further reduce the number of candidate objects and the number of I/O incurred. Then together with the Equation 4, we have the *Refinement* algorithm.

DOMINANCE BASED PRUNING.

We first introduce a *dominance* relationship between two objects regarding f, then present a *dominance based* pruning technique. Following is a formal definition of the *dominance* relationship between two objects.

Definition 5 (Dominance)**.** *For two objects U and V, if $f_\phi(V) < f_\phi(U)$ for any $\phi \in (0,1]$, then we say V dominates U regarding the preference function f, denoted by $V \prec_f U$.*

Example 11. *In Fig. 3, we have $A \prec_f C$ because $f_\phi(A) < f_\phi(C)$ for any $\phi \in (0,1]$. Since $f_{0.4}(A) > f_{0.4}(B)$, we have $A \not\prec_f B$.*

According to Definition 5, if $V \prec_f U$, we have $f_\phi(V) < f_\phi(U)$ for any $\phi \in (0,1]$. This implies $r(V) < r(U)$ according to Definition 4. The observation below indicates that an object U can be excluded from top k candidates if there are at least k other objects which *dominate* U regarding f.

Observation 1. *For a given set of objects $\mathcal{U}$ and a preferece function f, an object U can be eliminated from top k candidates if there exist k other objects which dominate U regarding f.*

Applying dominance based pruning. For two objects U and V, it is immediate that $V \prec_f U$ if $\Delta_{U,V} = 1$. We use U_{dc} to record the number of objects which *dominate* U. According to Observation 1, an object U can be eliminated from top k candidates if $U_{dc} \geq k$.

Using the aggregate information. Considering that it is expensive to access instances of an object because of the I/O cost incurred, we develop an I/O efficient approach to check *dominance* relationship of two objects; that is, when some pre-computed aggregate information of U is available, we may claim $V \prec_f U$ without accessing instances of U.

Let $\phi_{\gamma,f}(U)$ denote the total weights of the instances of U whose scores regarding f are not larger than γ; that is, $\phi_{\gamma,f}(U) = \sum_{u \in U, f(u) \leq \gamma} w(u)$. For instance, we have $\phi_{30,f}(C) = 0.8$ in Fig. 3. In order to avoid the computation of $\phi_{\gamma,f}(U)$ in which instances of U are accessed, we introduce a function $\bar{\phi}_{\gamma,f}(U)$ as an *upper bound* function of $\phi_{\gamma,f}(U)$, i.e., $\bar{\phi}_{\gamma,f}(U) \geq \phi_{\gamma,f}(U)$ for any score γ. Later, we will show how to derive $\bar{\phi}_{\gamma,f}(U)$ based on the aggregate information of U ,i.e., without accessing the instances of U.

Given two objects U and V, we suppose only V is loaded, i.e., the scores of the instances of V are computed regarding f. Lemma below implies that we can claim $V \prec_f U$ based on the *upper bound* function $\bar{\phi}_{\gamma,f}(U)$.

Lemma 1. *Given objects U and V, we have $V \prec_f U$ if $\bar{\phi}_{f^+(V),f}(U) < \phi_{f^-(U),f}(V)$.*

Proof: Let $\phi_0 = \phi_{f^+(V),f}(U)$ and $\phi_1 = \phi_{f^-(U),f}(V)$, as shown in Fig. 5, ϕ_0 and ϕ_1 correspond to the total weight of the instances in the shaded and striped areas respectively. $\phi_0 < \phi_1 = \phi_{f^-(U),f}(V)$ implies that $f_\phi(V) < f^-(U)$ for any $\phi \in (0, \phi_0]$. As $f_\phi(U) \geq f^-(U)$ always holds, we have $f_\phi(V) < f_\phi(U)$ for any $\phi \in (0, \phi_0]$. Since $f_\phi(V) \leq f^+(V)$ always holds and $f_\phi(U) > f^+(V)$ for any $\phi \in (\phi_0, 1]$, we have $f_\phi(V) < f_\phi(U)$ for any $\phi \in (\phi_0, 1]$. Therefore, $V \prec_f U$ holds according to Definition 5. Since $\bar{\phi}_{f^+(V),f}(U) \geq \phi_{f^+(V),f}(U) = \phi_0$, the correctness of the Lemma is immediate. □

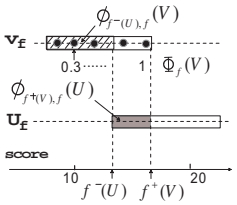


Fig. 5. Lemma 1

Below, we first introduce some aggregate information, then present a novel *dominance* checking approach based on the aggregate information and Lemma 1.

Definition 6 (*mean* ($\mu(U)$))**.** *We use $\mu(U)$ to denote the mean of an object U, where $\mu(U).D_i = \sum_{u \in U} u.D_i \times w(u)$.*

Definition 7 (*variance* $\sigma^2(U)$)**.** *$\sigma^2(U)$ denotes the variance of an object U on each dimension; that is, $\sigma_i^2(U) = \sum_{u \in U}(u.D_i - \mu(U).D_i)^2 \times w(u)$.*

Theorem below indicates that we can claim $V \prec_f U$ based on the aggregate information of U. Recall that, in Section 2 we assume $f(p) = \sum_{i=1}^{d} a_i \times p.D_i$ and $a_i > 0$ for $1 \leq i \leq d$.

Theorem 3. *Given two objects U and V, let $\gamma = f^+(V)$ with $f^-(U) \leq \gamma \leq f^+(U)$, we have $V \prec_f U$ if $\bar{\phi}_{\gamma,f}(U) < \phi_{f^-(U),f}(V)$, where $\bar{\phi}_{\gamma,f}(U) = \min_{i \in [1,d]}\{\varrho(\Delta_i, \sigma_i^2(U)\}$. Specifically, $\Delta_i = \mu(U).D_i - (l_i + \frac{\gamma - f^-(U)}{a_i})$, where l_i denotes the projection of the lower corner of U_{mbb} on i-th dimension and the function $\varrho(x,y)$ is defined as follows.*

$$\varrho(x,y) = \begin{cases} 1/(1+\frac{x^2}{y}) & \text{if } x > 0 \text{ and } y > 0 \\ 1.0 & \text{otherwise} \end{cases} \quad (5)$$

Please refer to Appendix A for details of the proof.

RANK BASED PRUNING.

Given two objects U and V, as $\Delta_{U,V}$ and $\Delta_{V,U}$ are computed at the same time, we do not need to compute $\Delta_{V,U}$ for V if $r(U)$ is already calculated. Instead, we keep a lower bound of the BC rank of an object V, denoted by $r^-(V)$. $r^-(V)$ may increase after the computation

of $r(U)$. For instance, suppose we first calculate $r(A)$ in Fig. 3, which results in $\Delta_{A,B} = 0.2$, $\Delta_{B,A} = 0.8$, $\Delta_{A,C} = 0$ and $\Delta_{C,A} = 1$. Then we have $r(A) = 0.2$, $r^-(B) = 0.8$ and $r^-(C) = 1$. Clearly, we can exclude B and C without further computation if $k = 1$, and hence the computation cost of $\Delta_{B,C}$ and $\Delta_{C,B}$ is saved.

Motivated by this, Theorem below indicates that an object U can be excluded from top k candidate if there exist k other objects whose BC ranks are higher than $r^-(U)$.

Theorem 4. *We can exclude an object U from top k candidates if $r^-(U) > \lambda_k$, where λ_k is the k-th highest BC rank computed so far.*

PUTTING TOGETHER.

Based on the pruning techniques proposed above, we present an efficient **Refinement** algorithm as illustrated in Algorithm 4. For each object $U \in \mathcal{S}$, $FD(U)$ and $PD(U)$ denote a set of objects, where $FD(U) = \{V | f^+(V) < f^-(U), V \in \mathcal{U}\}$ and $PD(U) = \{V | f^-(V) \leq f^+(U), V \in \mathcal{U} \setminus (U \cup FD(U))\}$ respectively. As shown in Fig. 4, we have $FD(B) = \{A\}$ and $PD(B) = \{C\}$. It is easy to see that $\Delta_{U,V} = 1$ for $V \in FD(U)$, and $\Delta_{U,V} = 0$ for $V \in \mathcal{U} \setminus (PD(U) \cap FD(U) \cap U)$. This implies that we do not need to explicitly compute $\Delta_{U,V}$ for $r(U)$ unless $V \in PD(U)$. Moreover, we have $V \prec_f U$ if $V \in FD(U)$ and hence Line 3 sets $r^-(U)$ and U_{dc} to $|FD(U)|$.

Line 5 processes *active* objects (e.g., top k candidates). As shown in Line 7, for each object $V \in PD(U)$ we do not need to compute $\Delta_{U,V}$ and $\Delta_{V,U}$ if V is processed at Line 5. This is because $\Delta_{U,V}$ is already computed since $V \in PD(U)$ implies $U \in PD(V)$. The Δ value computation can also be avoided if neither of two objects is an *active* object (i.e., top k candidates). Line 8 computes $\Delta_{U,V}$ and $\Delta_{V,U}$, then Line 9 updates $r^-(U)$ and $r^+(V)$. Note that when aggregate information of U is available, we have $U_{dc} = U_{dc} + 1$, $\Delta_{U,V} = 1$ and $\Delta_{V,U} = 0$ if $V \prec_f U$ holds according to Theorem 3; that is, the Δ values computation and the loading of instances of U may be avoided. We apply the pruning techniques to eliminate U from top k candidates in Lines 10-13. Specifically, λ_k is employed to keep the k-th highest BC rank seen so far for the *rank based* pruning. An object U can also be eliminated from top k candidates if $U_{dc} \geq k$ based on the *dominance based* pruning. In Line 15 we have $r(U) = r^-(U)$ if U remains *active*, and we also update λ_k based on $r(U)$.

Accessing Order. In Algorithms 4, we compute BC ranks of the objects in a *sequential* order. Intuitively, an object with high BC rank should be visited early for the effectiveness of the *dominance based* and *rank based* pruning techniques. Line 4 sorts the *active* objects in $\mathcal{S}$, where the key of an object U is $\frac{f^-(U)+f^+(U)}{2}$. Our empirical study shows that it performs better than other alternative keys such as $f^-(U)$ and $f^+(U)$. Note that when the aggregate information of the objects is available, the *active* objects are sorted based on $f(\mu(U))$ instead because intuitively the object with smaller $f(\mu(U))$ is more likely to be top k objects. This is confirmed by the empirical study.

Algorithm 4: ***Refinement***($\mathcal{S}$, f, k)

```
Input  : S : objects for refinement, f : preference
         function, k
Output : top k objects regarding f
1  λ_k := ∞;
2  for each U ∈ S do
3      r⁻(U) := |FD(U)|; U_dc := |FD(U)|;
4  L ← sort all active objects U ∈ S by (f⁻(U)+f⁺(U))/2;
5  for each active object U ∈ L visited in order do
6      for each V ∈ PD(U) do
7          if V is not processed and V or U is an active
           object then
8              ComputeDelta(U, V) ;
9              r⁻(U) := r⁻(U) + Δ_{U,V};
               r⁻(V) := r⁻(V) + Δ_{V,U};
10             if U_dc ≥ k or r⁻(U) > λ_k then
11                 U_stat:= non-active;
12             if V_dc ≥ k or r⁻(V) > λ_k then
13                 V_stat:= non-active;
14     if U_stat = active then
15         r(U) := r⁻(U); update λ_k ;
16 return k objects with highest BC ranks
```

Time Complexity. Let $|\mathcal{S}|$ denote the number of objects in $\mathcal{S}$, the time cost of Lines 2-3 is $O(|\mathcal{S}| \times \log(|\mathcal{S}|))$ because we can compute the $|FD(U)|$ for all object $U \in \mathcal{S}$ by sorting objects based on their f^+ value. It takes $O(|\mathcal{S}| \times \log(|\mathcal{S}|))$ time to sort objects in Line 4 to determine the access order. Suppose f^- and f^+ values of the objects are organized by an *interval tree* [9] with construction time $O(|\mathcal{S}| \times \log(|\mathcal{S}|))$, where $f^-(U)$ and $f^+(U)$ of an object U correspond to an *interval*, it takes $O(\log(|\mathcal{S}|) + |PD(U)|)$ time to retrieve $PD(U)$ in Line 6. Let n_Δ denote the total number of Δ value computations invoked in Line 8 and n_a be the number of objects loaded in Algorithm 4, the time cost in Lines 5-15 is $O(n_\Delta \times m + n_a \times m \log(m))$ because the instances of an object are sorted when they are loaded for Δ value computation (Line 8). Therefore, the time complexity of Algorithm 4 is $O(|\mathcal{S}| \times (\log(|\mathcal{S}|) + n_{ov}) + n_\Delta \times m + n_a \times m \log(m))$, where n_{ov} is the average size of $PD(U)$ for objects $U \in \mathcal{S}$. Although it remains $O(n^2 \times m)$ in the worst case, the performance of Algorithm 4 is very efficient in the empirical study because a large amount of objects are eliminated from the top k candidates and hence the CPU and I/O costs are significantly reduced. Note that Appendix B provides a comprehensive analysis on the I/O cost of Algorithm 4.

5 Extention and Future Work

In this Section, we discuss how to extend the techniques developed in this paper to other preference functions in Section 5.1 and Section 5.2. Section 5.3 discusses the possible future work of the paper.

5.1 General Linear Preference Functions

A more general linear preference function is in the form of $f(p) = \sum_{i=1}^{d} a_i \times p.D_i$, where a_i may be any real constant. All techniques in this paper can be immediately applied except that if $a_i < 0$, Equation 6 (instead of Equation 7) in Lemma 2 is used to derive the upper bound function $\bar{\phi}_{\gamma,f}(U)$ in Theorem 3 for i-th dimension.

5.2 Other Preference functions

Without considering the aggregate information, our techniques can be immediately applied to a preference function f if $f^-(U_{mbb})$ and $f^+(U_{mbb})$ can be calculated in a reasonable time for a given object U. Specifically, $f^-(U_{mbb}) = \min\{f(x)|x \in U_{mbb}\}$ and $f^+(U_{mbb}) = \max\{f(x)|x \in U_{mbb}\}$. This is not a demanding requirement because it takes $O(d)$ time for many sorts of preference functions, such as Eucledian distance function [29] and *monotonic* functions.

Clearly, Lemma 1 holds for any preference function. The main difference is how to derive $\bar{\phi}_{\gamma,f}(U)$ based on aggregate information of U regarding a preference function f. Below is a brief introduction of the motivation.

As shown in Fig. 6, the shaded area denotes the intersection of the space $\{x|f(x) \leq \gamma\}$ and U_{mbb}, total weight of the instances in this area equals $\phi_{\gamma,f}(U)$. Following the same rationale of Theorem 3, we can use the rectangle $r(a,d)$ or $r(a,b)$ to derive $\bar{\phi}_{\gamma,f}(U)$ based on $\mu(U)$ and $\sigma^2(U)$.

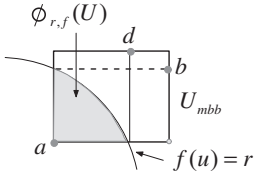


Fig. 6. Derive $\bar{\phi}_{\gamma,f}(U)$

5.3 Future Work

As discussed in Section 1, the main idea of the paper is to effectively and efficiently apply the rank aggregation methods to compute the final top k ranking results regarding all possible ϕ quantile where $\phi \in (0, 1]$. Therefore, theoretically, any rank aggregation approach which does not need training can be applied. Nevertheless, from a technique point of view, the challenges are twofold. Firstly, the number of possible ϕ value is infinite, and hence we need to reduce the search space. Secondly, as the number of possible ϕ values is still very large after search space reduction, it is essential to develop efficient rank computation algorithm.

In the paper, we adopt the Borda Count method because we can effectively address above two issues. Moreover, as shown in [11] and [28], the Borda Count method is time efficient, and it satisfies some important properties such as *anonymity*, *neutrality* and *consistency*.

City Names	*Avg-R*	*Med-R*	*BC*
Richmond, VA	1 (72.83K)	9 (47.52K)	6
Greensboro, NC	2 (69.62K)	8 (47.59K)	4
Chattanooga, TN	3 (68.36K)	3 (47.88K)	3
Pittsburgh, PA	4 (68.37K)	7 (48.6K)	7
Montgomery, AL	5 (62.40K)	4 (47.87K)	2
Philadelphia, PA	6 (62.36K)	6 (47.92K)	9
Pasadena, TX	7 (61.39K)	2 (49.07K)	5
Newport News, VA	8 (60.41K)	10 (47.34K)	8
Vancouver, WA	9 (58.13K)	5 (48.19K)	10
Hampton, VA	10 (58.07K)	1 (50.92K)	1

TABLE 2
Evaluating Cities

However, the Borda Count method cannot satisfy the Condorcet criterion [28], [11]. Therefore, it is worthwhile to investigate how to effectively apply other more sophisticated rank aggregation methods such as footrule method and markov train method [11]. The main challenge is how to develop efficient rank computation algorithms. In BC rank based algorithm, we can compute the BC rank of an object by pair-wise computions with other objects , and hence develop efficient *pruning* techniques which can effectively bound the BC rank of an object during the computation (i.e., without accessing all other objects). However, it is far from trivial to develop similar *pruning* techniques for the algorithms based on the footrule method and markov chain method due to the higher complexity of the footrule method and markov chain method.

6 Performance Evaluation

In this section, we evaluate the effectiveness of BC Rank, and the efficiency and scalability of our methods.

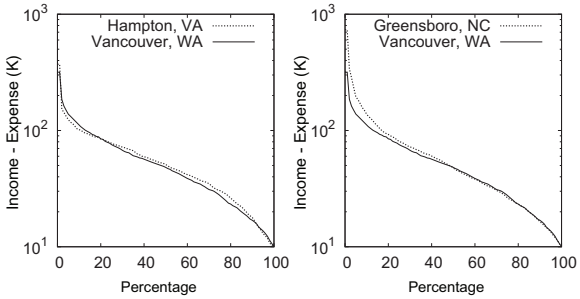
6.1 Effectiveness of BC Rank

We use the 2009 Integrated Public Use Microdata (http://usa.ipums.org/usa, IPUMS for short) and the paper citation dataset (http://www.arnetminer.org/citation) to evaluate the ranking approaches. In the IPUMS dataset, a city is a *multi-valued* object and an instance corresponds to a record of an individual in the city with two attributes: family income and expense. In the paper citation dataset, an author is treated as a *multi-valued* object, an instance is the number of citations of his/her individual paper published, and the number of instances is the number of papers by the author. As discussed in Section 1, we do not compare with the probabilistic based approaches due to the inherent difference between uncertain data model and *multi-valued* object model. We assume all instances of an object carry the same weight, and use the `average` and `median` as the representatives for the majority based rankings. Note that `median` is a special case of ϕ-quantile where $\phi = 0.5$.

6.1.1 Results on the IPUMS Data

We use $income$-$expense$ (i.e., $discretionary\ income$) as the scoring function to rank the livability of 10 cities where the average discretionary income is around 60K. For avg-based ranking (AVG-R) and median-based ranking (Med-R), we rank cities based on their $average\ discretionary\ income$ and $median\ discretionary\ income$ respectively. Table 2 shows the ranking of the cities based on avg-based ranking (AVG-R), median-based ranking (Med-R) and the BC ranking (BC). Although there are several well studied methods to compare two income distributions, such as stochastic dominance [15], those methods generate partial orders instead of a ranking, and thus cannot be used to answer ranking queries.

As shown in Table 2, the three rankings are quite different. For instance, Richmond has the highest average discretionary income, but is ranked number 9 and 6 by the median and BC ranking methods. While Hampton is ranked numbers 10, 1 and 1, respectively, by the avg-based ranking, median-based ranking and BC ranking. To compare the effectiveness of different ranking approaches, Fig. 7(a) and Fig. 7(b) depict the discretionary income distributions of three cities, in which families are ordered by their discretionary incomes and x axis is in percentage of the families. Fig. 7(a) shows discretionary income distributions of Vancouver and Hampton. The rich families in Vancouver (top-20%) have higher discretionary income than those in Hampton, and also bring up the average discretionary income of the city. However, most of the lower-80% families in Hampton have a higher discretionary income than those in Vancouver and hence is ranked higher by BC ranking and median-based ranking. Fig. 7(b) illustrates discretionary income distributions of Vancouver and Greensboro. The upper-45% families in Greensboro have significant higher discretionary income than those in Vancouver, while the discretionary income of two cities become quite similar in the remaining part. It is intuitive that Greensboro should be ranked higher than Vancouver, which is captured by BC ranking and avg-based ranking. But median-based ranking is in favor of Vancouver because the discretionary income of the 50% families are ranked slightly higher than those in Greensboro. The results in two Fig. 7(a) and Fig. 7(b) clearly show that the BC ranking makes a more comprehensive and unbiased comparison than the avg-based ranking and median-based ranking.



(a) Vancouver vs Hampton (b) Vancouver vs Greensboro

Fig. 7. Score Distributions

6.1.2 Results on the Paper Citation Data

The h-index (the maximum value h such that an author has h papers each cited at least h times) is a popular method used to indicate the impact of an author. Therefore, we use h-index based ranking as a baseline to evaluate the effectiveness of avg-based ranking and BC ranking on paper citation data. Table 3 shows the rankings of 10 well known authors regarding avg-based ranking (AVG-R), BC ranking (BC), and h-index based ranking (H-R), respectively. In avg-based ranking, median-based ranking and h index based ranking, authors are ranked based on their average number of citations, the meadian of the number of citations and h index value (h-index) respectively, which are shown in the bracket next to their corresponding ranks.

Compared with the avg-based ranking and median-based ranking, the BC ranking is more consistent with the h-index based ranking. For instance, Massoud Pedram has the largest h-index value among the 10 authors. He is ranked numbers 1, 6 and 6 in the BC ranking, avg-based ranking and the median-based ranking, respectively. The BC ranking is superior to the avg-based and median-based rank methods as shown in the example, since it produces rankings closer to the h-index based ranking that is popularly used in the domain. Note that h-index is not a general ranking approach. For instance, we cannot use h-index to rank the IPUMS dataset.

Author Names	Avg-R	Med-R	BC	H-R
Rami G. Melhem	1 (26.70)	4 (9)	3	3 (40)
Tzi-cker Chiueh	2 (26.67)	3 (10)	2	4 (35)
Hong-Yuan Mark Liao	3 (26.65)	10 (3)	9	8 (27)
Ronald R. Yager	4 (26.64)	4 (9)	4	2 (42)
Yu-Kwong Kwok	5 (26.63)	7 (5)	8	10 (24)
Massoud Pedram	6 (26.60)	6 (7)	1	1 (50)
Norbert Fuhr	7 (26.43)	1 (11)	5	5 (33)
Mark Sanderson	8 (26.40)	7 (5)	7	7 (28)
Dewayne E. Perry	9 (26.33)	9 (4)	10	8 (27)
Olivier Danvy	10 (26.21)	1 (11)	6	6 (31)

TABLE 3
Evaluating Authors

Discussion 1. *In the above experiments, we show that the BC ranking method can provide a comprehensive ranking result since it can capture the score distributions of the objects. Nevertheless, we can also construct some examples in which the BC ranking method is outperformed by avg-based and quantile-based ranking methods. For instance, suppose objects A and B have 9 instances each, and instances have the same weight. The instances of each object are sorted by their scores in decreasing order. Assuming the larger score*

value is preferred, we have that $f(a_i)$ is much smaller than $f(b_i)$ for $1 \leq i \leq 4$ and $f(a_i)$ is slightly larger than $f(b_i)$ for $5 \leq i \leq 9$. In this example, A is ranked first regarding BC ranking approach while B wins out for the `avg`-based method and the `quantile`-based ranking method where $\phi = 0.3$. Intuitively, B should be the winner if only A and B are involved in the comparison because B significantly outperforms A regarding a considerable number of quantile values. However, we do not observe this kind of score distributions in our empirical study. Moreover, if A and B are evaluated together with other objects, B is likely to win regarding BC ranking method because B is ranked much higher than A when $1 \leq i \leq 4$.

6.2 Efficiency and I/O Model Accuracy

Now, we evaluate the efficiency and scalability of our techniques developed in this paper.

6.2.1 Settings and Data Sets

We evaluate the following algorithms.

- **QB** The Quantile based Algorithm (Algorithm 1) presented in Section 3.
- **QB-S** First apply the *filtering* technique (Algorithm 3) proposed in Section 4, then run Algorithm QB. According to Theorem 2, it suffices for QB Algorithm to retrieve top k answer based on the objects survived from filtering.
- **PB** The index based Algorithm (Algorithm 2) proposed in Section 4 **without** aggregate information.
- **PB-A** The index based Algorithm (Algorithm 2) proposed in Section 4 **with** aggregate information .

All algorithms are implemented in standard C++ with STL library support and compiled with GNU GCC. Experiments are run on a PC with Intel Xeon 2.40GHz dual CPU and 4G memory running Debian Linux. The disk page size is set to 4096 bytes. In the experiments, the MBBs of the objects are organized by R-Tree. When aggregate information is used, the aggregate information (*mean* and *variance*) of an object is kept with its MBB in the the argument R-Tree.

Our experiments are conducted on both synthetic datasets and real dataset.

To generate a synthetic data set, we first generate the centres of n objects using the benchmark data generator in [3], where n varies from $20K$ to $100K$ with the default value $20,000$. *Anti-correlated* (A), *Correlated* (C) and Independent (E) distributions of n object centres are used in our experiments. By default, we use *Anti-correlated* distribution. Then, for each object we create a hyper-rectangle region where the instances of this object appear. The edge size of the hyper-rectangle region follows a uniform distribution in range $(0, 4r]$ with expectation $2r$, where r varies from 100 to 400 with default value 200.

The number of instances of an object follows a uniform distribution in range $[1, 2m]$ where m varies from 200 to $1,000$ with the default value 400. In expectation, each object has m instances and the total number of instances in a dataset is nm; by default, it is $8,000,000$. Instances of an object may follow four distributions *Uniform* (U), *Normal* (N), *Zipf* (Z) and *Mixed* (M). In the first two distributions, instances of an object follow *Uniform* or *Constrained Normal* distribution and the standard deviation (σ) of *Constrained Normal* is set to $0.2 \times r$. For *Zipf* distribution, we first randomly choose an instance u, then the distances of other instances to u follow the *Zipf* distribution with $z = 0.5$. The instance of *Mixed* distribution randomly chosen from one of three distributions. In this paper, all instances of an object have the same weight.

For the distributions of object centres and the instances within an object, we have Anti-Mixed datasets where the centers of object MBBs follow Anti-correlated distribution, while instances follow *Mixed* distribution, denoted by A_M. Similarly, we have A_U, A_N, A_Z, E_M, E_N and C_M datasets. A_M is used by default in the experiments. The dimensionalities of the datasets vary from 2 to 5, with default value 3.

We use the NBA game-by-game technique statistics from 1991 to 2005. The NBA dataset is downloaded from www.nba.com and consists of $339,721$ records (instances) of $1,313$ players. We treat each player as an object and the records of a player as the instances of the corresponding object. Instances of an object take equal weight. Three attributes are selected in our experiments: the number of points, the number of assistants, and the number of rebounds. The larger the attribute values, the better.

Moreover, we generate two datasets in which the centers of the objects are obtained from the Forest CoverType dataset (FC)[1] and Household dataset [2]. In FC, we select the horizontal distances of each observation point to the Hydrology and roadways. In Household, each record represents the percentage of an American familys annual income spent on 3 types of expenditures (e.g., gas, etc.). The instances of two datasets follow the *Mixed* distribution. There are $581,012$ and $127,932$ objects in FC and Household respectively. And the total number of instances in FC and Household are $116.6m$ and $51.2m$ respectively.

In the experiments, a workload consists of 500 linear functions where $f(p) = \sum_{i=1}^{d} a_i \times p.D_i$. We randomly choose a_i from $(0, 1]$ for each dimension. In the paper, the average query processing time, which includes the CPU time and I/O latency, is used to measure the performance of the algorithms. Moreover, we also record the number of I/Os incurred for the algorithms.

Table 4 below lists parameters which may potentially have an impact on our performance study. In the experiments, all parameters use default values unless otherwise specified.

1. http://archive.ics.uci.edu/ml/datasets.html
2. http://www.ipums.org

Notation	Definition (Default Values)
r	avg. edge length (200)
n	the number of uncertain object ($20K$)
k	k value for the ranked query (40)
m	avg. number of instances of each object(400)
d	dimensionality (3)

TABLE 4
System Parameters

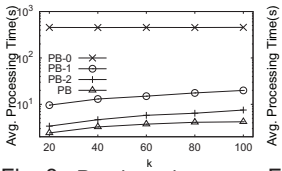


Fig. 8. Pruning rules

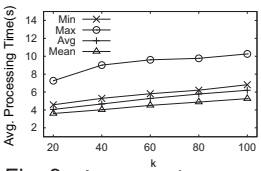


Fig. 9. Access orders

6.2.2 *Efficiency*

We evaluate the effectiveness of the filtering and pruning techniques against the default dataset. The *filtering* technique, *dominance based* pruning technique and *rank based* pruning technique, are incrementally included in $PB1$, $PB2$ and PB. Note that $PB0$ does not have filtering and pruning techniques. As shown in Fig. 8, the filtering and pruning techniques significantly reduce the query processing time as a large number of objects are eliminated from top k candidates. Fig. 9 investigates the impact of the accessing order in Algorithm 2 against the NBA dataset, where the min, max , avg and *mean* represent that the key of an object U is $f^-(U)$, $f^+(U)$, $\frac{f^-(U)+f^+(U)}{2}$ and $f(\mu(U))$ respectively. As expected, the *average* score based approach outperforms the *minimal* and *maximal* score based ones, and the *mean* based method can further improve the performance.

We evaluate the performance of the algorithms against datasets A_M, A_U, A_N, A_Z, E_M C_M, NBA, FC and $Household$ in Fig. 10 where the empty bars on top of each algorithm represent the CPU costs, while the bars below stand for the I/O latency. As expected, the performance of QB is always ranked the last among four algorithms. QB-S achieves a magnitude of speed up on synthetic datasets by applying the filtering technique. PB outperforms QB by at least one magnitude on all datasets, and PB-A further improves the CPU and I/O cost by taking advantage of the aggregate information. Note that the improvement of I/O cost is not significant on NBA dataset because the MBBs of the objects are heavily overlapped and hence many objects are loaded. Nevertheless, it is shown that the CPU cost reduced by our pruning techniques is significant. We excluded QB Algorithm from the following experiments for better evaluation of other algorithms.

We study the impact of k on the query processing time and I/O costs of Algorithms QB-S, PB and PB-A in Fig. 11. The performance are evaluated on the default dataset (A_M) and FC. It is reported that the average processing time and I/O costs of three algorithms increase slowly against the growth of k. Same as Fig. 10, PB significantly outperforms QB-S, and PB-A further reduces the processing time and I/O cost.

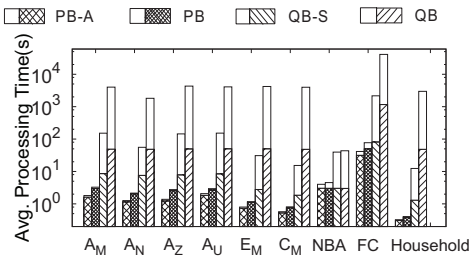
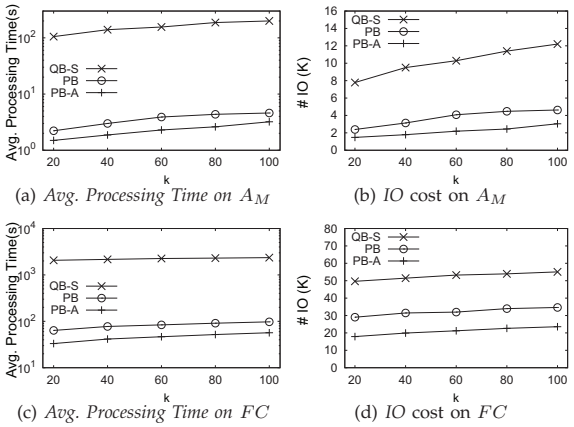


Fig. 10. Different datasets



(a) *Avg. Processing Time on* A_M (b) *IO* cost on A_M

(c) *Avg. Processing Time on* FC (d) *IO* cost on FC

Fig. 11. The effect of k

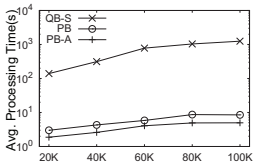


Fig. 12. Diff. n

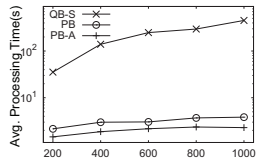


Fig. 13. Diff. m

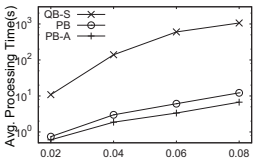


Fig. 14. Diff. r

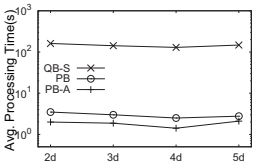


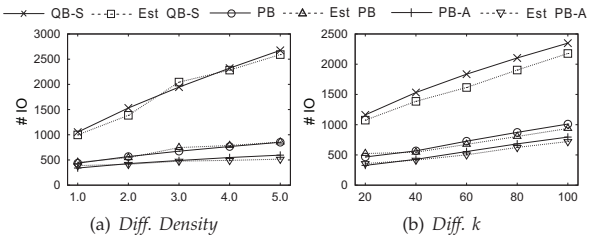
Fig. 15. Diff. d

We further investigate the impact of the number of objects, the number of instances, the edge length and the dimensionality against the performance of Algorithms QB-S, PB and PB-A. As expected, the performance of Algorithms PB and PB-A degrades on the growth of n and m in Fig. 12 and Fig. 13 respectively. Nevertheless,

the increase of their costs is slow compared with QB-S. Fig. 14 demonstrates the performance of three algorithms also degrades on the growth of r. It is interesting that the performance of three algorithms influctuates against the growth of the dimensionality in Fig. 15. This is because the degree of the MBBs' overlapping is reduced when the dimensionality increases with same edge length, which reduces the number of objects involved in the computation, but on the other hand the computational cost and I/O cost for each object are more expensive when the dimensionality is high.

6.2.3 Cost Model Accuracy

We evaluate the accuracy of the cost model proposed in Section B for Algorithms QB-S, PB and PB-A against dataset E_N. Recall that E_N denote the synthetic dataset where the centers of the objects follow the independent (E) distribution and the instances of each object follow the *Normal* (N) distribution. Fig. 16 plots the I/O comparison as a function of the *density* and k respectively. The proposed cost model is highly accurate, incurring a *relative error* less than 7% in most of the cases. As expected, PB-A achieves the best I/O performance in all settings.



(a) *Diff. Density* (b) *Diff. k*

Fig. 16. I/O cost estimation

7 Related Work

In this section, we introduce the existing works which are closely related to the paper.

7.1 Conventional top k computation

Top k query processing is a very active research area and various novel techniques have been developed. For instance, FA [12] and TA [21] algorithms are proposed to efficiently compute the top k objects to reduce the access cost, where the attribute values of each object are sorted and resident on multiple (distributed) data repositories. Many efficient index techniques have been proposed to pre-compute and organize the candidate set based on the geometric properties of the convex and skyline, such as *Onion* [5] and Skyband [20]. However, in those conventional top k query processing techniques, an object is usually represented by a point in a multidimensional space and hence is ranked based on its unique score regarding a preference function. Therefore, those techniques cannot be directly applied to top k query on *multi-valued* objects because the multiple instances of a *multi-valued* object lead to multiple scores.

7.2 Top k query for Uncertain Objects

Recently, a large amount of work has been dedicated to top k queries on uncertain objects with different semantics, such as U-top k [25], U-k ranks [25], expected rank based top k [7], [16] and quantile rank based top k [16], parameterized ranking function based top k [17].

7.3 Rank Aggregation Methods

The problem of rank aggregation [4], [1], [11], [22] is to compute a consensus ranking, given a set of individual ranking preferences. For instance, in the metasearch the system need to combine the results returned by multiple search engines in response to a given query. As shown in the seminal work by Dwork in [11], there are many rank aggregation methods such as Borda Count method, Footrule methods and Markov chain methods. Borda Count is usually described as a consensus based voting system and has a wide spectrum of applications. In order to avoid the "tyranny of the majority" [13] which may happen in majority based voting systems, the Borda's method can seek the consent, not necessarily the agreement, of participants and the resolution of objections, which is widely used in various decision systems [2], [27]. It is shown in [11] that the footrule optimal aggregation is a good approximation of Kemeny optimal aggregation, which ensures the satisfaction of the extended Condorcet principle [24]. The markov chain methods are proposed in [11], which can provide a more holistic viewpoint of comparing all candidates against each other.

7.4 KNN Query on *multi-valued* Objects

Zhang *et al.* [29] present a relevant study on *multi-valued* objects, where the quantile distance is employed to retrieve k nearest neighbors. The key idea is to use ϕ-quantile of the distance (score) distribution of two *multi-valued* objects to measure their closeness, where ϕ ($\phi \in (0, 1]$) is predefined. The ϕ-quantile based ranking is equivalent to `median`-based rankind when $\phi = 0.5$. Similar to `median`-based ranking, for a pregiven ϕ value, the ϕ-quantile based approach cannot properly capture the distribution.

8 Conclusions

To effectively and efficiently process top k queries on *multi-valued* objects, which has several applications, we propose a novel ranking semantics, BC rank. We develop effective and efficient BC rank based top k algorithms and conduct comprehensive experiments on both real and synthetic data to demonstrate the effectiveness and efficiency of our techniques.

REFERENCES

[1] J. A. Aslam and M. H. Montague. Models for metasearch. In *SIGIR*, 2001.
[2] J. A. Benediktsson and I. Kanellopoulos. Classification of multisource and hyperspectral data based on decision. *IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING*, 1999.
[3] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE 2001*.
[4] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR*, pages 21–28, 1995.
[5] Y.-C. Chang, L. D. Bergman, V. Castelli, C.-S. Li, M.-L. Lo, and J. R. Smith. The onion technique: Indexing for linear optimization queries. In *SIGMOD*, 2000.
[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (second edition)*. 2001.
[7] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, 2009.
[8] G. Das, D. Gunopulos, N. Koudas, and N. Sarkas. Ad-hoc top-k query answering for data streams. In *VLDB*, 2007.
[9] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications (2nd edition)*. Springer-Verlag, 2000.
[10] J.-C. de Borda. *Mmoire sur les lections au scrutin*. 1781.
[11] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.
[12] R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1), 1999.
[13] L. Guinier. *Tyranny of the Majority : Fundamental Fairness in Representative Democracy*. 1995.
[14] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference 1984*.
[15] R. Herrerias-pleguezuelo. *Distribution Models Theory*. 2005.
[16] J. Jestes, G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data. *IEEE Trans. Knowl. Data Eng.*, 23(12):1903–1917, 2011.
[17] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1), 2009.
[18] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *SIGMOD Conference*, pages 426–435, 1998.
[19] R. Meester. *A Natural Introduction to Probability Theory*. 2004.
[20] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In *SIGMOD*, 2006.
[21] S. Nepal and M. V. Ramakrishna. Query processing issues in image (multimedia) databases. In *ICDE*, 1999.
[22] M. E. Renda and U. Straccia. Web metasearch: Rank vs. score based rank aggregation methods. In *SAC*, 2003.
[23] A. D. Sarma, O. Benjelloun, A. Y. Halevy, S. U. Nabar, and J. Widom. Representing uncertain data: models, properties, and algorithms. *VLDB J.*, 2009.
[24] J. H. Smith. Aggregation of preferences with variable electorate. *Econometrica*, 41(6):pp. 1027–1041.
[25] M. A. Soliman, I. F. Ilyas, and K. C. Chang. Top-*k* query processing in uncertain databases. In *ICDE 2007*.
[26] Y. Theodoridis and T. K. Sellis. A model for the prediction of r-tree performance. In *PODS*, pages 161–171, 1996.
[27] A. P. Topchy, M. H. C. Law, A. K. Jain, and A. L. N. Fred. Analysis of consensus partition in cluster ensemble. In *ICDM*, 2004.
[28] H. P. Young. An axiomatization of borda's rule. *Journal of Economic Theory*, 9(1):43–52, 1974.
[29] W. Zhang, X. Lin, M. A. Cheema, Y. Zhang, and W. Wang. Quantile-based knn over multi-valued objects. In *ICDE*, 2010.

Ying Zhang is a Research Fellow and Australian Research Council Australian Postdoctoral Fellowship (ARC APD) holder (2010-2013) in the School of Computer Science and Engineering, the University of New South Wales. He received his BSc and MSc degrees in Computer Science from Peking University, and PhD in Computer Science from the University of New South Wales. His research interests include query processing on data stream, uncertain data and graphs.

Wenjie Zhang is currently a lecturer and ARC DECRA (Australian Research Council Discovery Early Career Researcher Award) Fellow in School of Computer Science and Engineering, the University of New South Wales, Australia. She received PhD in computer science and engineering in 2010 from the University of New South Wales. Since 2008, she has published more than 20 papers in SIGMOD, VLDB, ICDE, TODS, TKDE and VLDBJ. She is the recipient of Best (Student) Paper Award of National DataBase Conference of China 2006, APWebWAIM 2009, Australasian Database Conference 2010 and DASFAA 2012, and also co-authored one of the best papers in ICDE2010, ICDE 2012 and DASFAA 2012. In 2011, she received the Australian Research Council Discovery Early Career Researcher Award.

Jian Pei is a Professor at the School of Computing Science, Simon Fraser University, Canada. He is interested in researching, developing, and deploying effective and efficient data analysis techniques for novel data intensive applications, including data mining, Web search, data warehousing and online analytic processing, database systems, and their applications in social networks and media, health-informatics, business and bioinformatics. He has published prolifically and his publications have been cited thousands of times. He has served regularly for the leading academic journals and conferences in his fields. He is an ACM Distinguished Speaker, and a senior member of ACM and IEEE. He is the recipient of several prestigious awards.

Xuemin Lin is a Professor in the School of Computer Science and Engineering, the University of New South Wales. He has been the head of database research group at UNSW since 2002. He is a concurrent professor in the School of Software, East China Normal University. Before joining UNSW, Xuemin held various academic positions at the University of Queensland and the University of Western Australia. Dr. Lin got his PhD in Computer Science from the University of Queensland in 1992 and his BSc in Applied Math from Fudan University in 1984. During 1984-1988, he studied for Ph.D. in Applied Math at Fudan University. He currently is an associate editor of ACM Transactions on Database Systems. He is a senior member of IEEE. His current research interests lie in data streams, approximate query processing, spatial data analysis, and graph visualization.

Qianlu Lin is currently a PhD student in the School of Computer Science and Engineering, the University of New South Wales, Australia. She received her B.S. degree in computer science from the University of New South Wales, Australia. Her research focuses on high-dimensional indexing.

Aiping Li received the BS and PhD degrees in computer science and technology at National University of Defense Technology (NUDT), China in 2000 and 2004, respectively. He has been an associate professor in the Institute of Computer and Technology of National University of Defense Technology, Since 2006. He visited the University of New South Wales, Australia. His research interests include uncertain databases, data mining,spatial databases and time series databases.He is a member of the IEEE.