Journal of Computer and System Sciences 89 (2017) 92–113

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Efficient monochromatic and bichromatic probabilistic reverse top-*k* query processing for uncertain big data

Guoqing Xiao [a], Kenli Li [a,b,*], Xu Zhou [a], Keqin Li [a,b,c]

[a] College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China
[b] National Supercomputing Center in Changsha, Changsha 410082, Hunan, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 7 March 2016
Received in revised form 20 May 2016
Accepted 31 May 2016
Available online 16 July 2016

Keywords:
Big data
Data management
Probabilistic reverse top-*k* queries
Query processing
Uncertain data

ABSTRACT

There has been an increasing growth in numerous applications that naturally generate large volumes of uncertain data. By the advent of such applications, the support of advanced analysis query processing such as the top-*k* and reverse top-*k* for uncertain big data has become important. In this paper, we model firstly probabilistic reverse top-*k* queries over uncertain big data for the discrete situation, in both *monochromatic* and *bichromatic* cases, denoted by MPRT and BPRT queries, respectively. We determine the partitions of solution space of MPRT queries and provide in theory a mathematical model for solving arbitrary dimensional data space. Additionally, we propose effective pruning heuristics to reduce the search space of BPRT queries. Moreover, efficient query procedures are presented seamlessly with integration of the proposed pruning strategies. Extensive experiments demonstrate the efficiency and effectiveness of our proposed approaches with various experimental settings.

1. Introduction

Recently, the support of rank-aware query processing has played an increasingly important role in many real-world applications, such as market analysis, multi-criteria decision making, environmental surveillance, quantitative economics research, etc. This has created a need for data management algorithms and applications, among which a pivotal technique in this respect is the top-*k* query, which is a classic problem in the area of databases and information retrieval. Top-*k* query retrieves only the *k* objects that best match the user preferences based on a weighting function, thus avoiding huge and overwhelming answer sets [11]. It is very important for a manufacturer that its products are returned in the highest ranked positions for as many different user preferences as possible. On the other hand, a reverse top-*k* query for a product returns a set of users (or customers), named potential users, who regard the product as one of their top-*k* result sets [39]. A product in the top-*k* answer set of a user implies that the product meets the preference of the user more than the products not in the top-*k* result set. Accordingly, the number of potential users in the reverse top-*k* answer set for a product becomes a good estimate of the popularity of the product in the market.

Tables 1 and 2 illustrate an example of top-*k* queries and reverse top-*k* queries over data collected from a real estate deal database containing information about different houses as well as user preferences. For each of the three houses, the price and area are recorded, and smaller value on each attribute is more preferable. The database also stores the preferences of

* Corresponding author at: College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China.
E-mail addresses: xiaoguoqing@hnu.edu.cn (G. Xiao), lkl@hnu.edu.cn (K. Li), zhouxu@126.com (X. Zhou), lik@newpaltz.edu (K. Li).

http://dx.doi.org/10.1016/j.jcss.2016.05.010
0022-0000/

Table 1
A product data set and its reverse top-2 results.

House	Price	Area	Prob.	Reverse top-2 results
a	0.8	0.1	0.6	{Alice, Chris}
b	0.2	0.7	0.7	{Bob, Chris}
c	0.5	0.5	0.5	{Alice, Bob}

Table 2
A user preference data set and its top-2 results.

User	Preference		
	μ [price]	μ [area]	Top-2 results
Alice	0.1	0.9	{*c*, *a*}
Bob	0.8	0.2	{*b*, *c*}
Chris	0.5	0.5	{*a*, *b*}

three users (Alice, Bob, and Chris) in terms of weights on the attributes. Different users can have various preferences about a potential house. For example, Alice prefers houses with spacious room, whereas Bob is interested in cheap houses. Chris is indifferent or values equally price and area. In order to recommend the most popular houses to users, the brokers need to know the potential user community of each house property. This can be implemented by executing a reverse top-k query. As illustrated in bold in Table 1, the reverse top-2 result set, i.e., the two potential users, for house property a, b, and c are {Alice, Chris}, {Bob, Chris}, and {Alice, Bob}, respectively. Furthermore, in Table 2, the top-2 houses are described in bold for each user along with their aggregate scores.

In addition, lots of other real-life applications, such as e-commence and online transactions etc., can be taken as the motivating examples to model the uncertain/probabilistic databases. For example,

(1) A motivating application can be the advertisement for e-commerce since there are a lot of identical items, the agency maintains an item database which is available for buyers (users) in the market. Additionally, the agency regularly collects the current preferences of the buyers (users) for favored characteristics of items, as determined by the market demand. In order to facilitate a beneficial strategy for the promotion of an item, the agency needs to identify the most influential items in the market such that the agents can prioritize showing such items to users. Reverse top-k queries can be applied to find which l items should be on advertisement.
(2) In a stock online exchange system, the bourse is interested in finding l stocks which have the largest investment potential based on historical stock market information and trader preferences, in order to perform direct marketing to its investor in a timely fashion. The l stocks form the most influential stocks of the bourse.

However, in reality, the items for advertising in the Internet and the stocks in a stock-trading system, due to data collecting, transmission delay, and inaccuracy or incompletion in conversion, etc., each item/stock can be regarded as an uncertain data record and can be given a definite probability value for characterizing its confidence or authenticity [2]; likewise, in a rent/sale system of house properties, each property is associated with a trustability value which is derived from customers feedback on the property quality, surrounding environment and traffic conditions, etc. This trustability value can also be regarded as existence probability of the property since it represents the probability that the house occurs exactly as described in the advertisement in terms of quality and surrounding environment [49]. Thus the trustability value is an important factor whether the users rent/purchase the house. In general, this kind of databases can be modeled as an uncertain/probabilistic database. For instance, in Table 1, the probability column (prob.) shows the trustability of each house property in the market.

As a matter of fact, due to data noise, transmission delay, environmental factors, and so on, the uncertainty of data is existent in essence. Similarly, surveys and imputation techniques create data which is uncertain in nature. There are numerous other examples which demonstrate the existences of uncertainty in the collected data, such as sensing data [12,1,27], moving object search [5,18,52], location-based services (LBS) [6,34,4], etc. These applications have created a need for uncertain big data management algorithms, among which a pivotal technique in this respect is the query processing over uncertain databases, such as top-k queries and reverse top-k queries. With the rapid development of data collection methods and the practical applications, the issue of uncertain data query has drawn wide attention in both academia and industry.

Although previous works [39,42,40,41,43] have studied the reverse top-k search problem over precise data, they cannot directly handle the uncertain databases. Jin et al. [24] proposed firstly reverse top-k queries upon uncertain databases. However, in this work, each object in the uncertain database is described as a probabilistic distribution function (i.e., the continuous situation) rather than a vector (i.e., the discrete situation) which is more practical for describing the attributes of an object and the preferences of a user. Thus, the present work cannot handle the discrete case directly. We have to redefine new query semantics for probabilistic reverse top-k queries.

In this paper, we define the probabilistic reverse top-k query in the context of uncertain databases. At first, a kind of query, namely *monochromatic probabilistic reverse top-k* (MPRT) query, is proposed. It retrieves all potential users whose uncertain top-k results contain a given query object with its top-k probability greater than or equal to a user-specified probability threshold. In this type of query, there is no knowledge of user preferences and a manufacturer aims to obtain all potential users for a given product and evaluate the impact that the product would have on the market. For instance, in our illustrated example of 2-dimensional Euclidean space, as shown in Section 4.1, given a probability threshold 0.5, we know that the solution space of the monochromatic probabilistic reverse top-1 of House a is $\mathcal{U}_1 = \mathcal{U}_{11} \cup \mathcal{U}_{12}$. Correspondingly, the impact of House a being ranked in the 1st in the market can be estimated as $IF_a^1 = \mathscr{S}_1/\mathscr{V} = (\mathscr{S}_{11} + \mathscr{S}_{12})/\mathscr{V} = 17\%$, where $\mathscr{S}_1$, $\mathscr{S}_{11}$, and $\mathscr{S}_{12}$ are the area of planes $\mathcal{U}_1$, $\mathcal{U}_{11}$, and $\mathcal{U}_{12}$, respectively, $\mathscr{V}$ is the area of whole plane $\mathcal{U}$. Likewise, the impacts of a being ranked in the 2nd and the 3rd in the market are $IF_a^2 = 17.51\%$ and $IF_a^3 = 35.88\%$, respectively.

Apart from the MPRT query, furthermore, we also study the *bichromatic probabilistic reverse top-k* (BPRT) query which is more useful in real applications, involving two distinct data sets. Specifically, given an uncertain object set $\mathcal{D}$, a user preference set $\mathcal{U}$, and an uncertain query object q in $\mathcal{D}$, a BPRT query returns those users $\mu \in \mathcal{U}$ such that the uncertain top-k results based on μ in $\mathcal{D}$ contain q. Since the retrieval of BPRT query requires examining the entire data set(s) to compute the top-k probabilities of uncertain objects for each user, it is quite inefficient and even infeasible. Moreover, with the rise of big data, the scale of this problem has increased dramatically, and computations of such operators are challenging today since there is an increasing trend of applications to deal with uncertain big data. In particular, due to the existence of probability dimension, uncertain data query is processed in a possible world space which grows exponentially. Thus, several efficient pruning heuristics with the pruning ratio around 95% are proposed for the BPRT query to reduce its search space.

In summary, we make the following significant contributions in this paper.

- We formalize two queries, including *monochromatic* and *bichromatic probabilistic reverse top-k* (MPRT and BPRT, respectively) queries, over uncertain databases. To the best of our knowledge, this is the first time that probabilistic reverse top-k queries are proposed for the discrete situation, considering both monochromatic and bichromatic cases.
- We determine the partitions of solution space of MPRT queries and provide in theory a mathematical model for solving arbitrary dimensional data space.
- We propose effective pruning heuristics, which are important and efficient for query processing of uncertain big data, to help reduce the search space for BPRT queries. The proposed pruning methods can be seamlessly integrated into our query procedures and efficiently retrieve the query results.
- Last but not least, we demonstrate through extensive experiments the effectiveness of our pruning methods as well as the efficiency of our algorithms under various experimental settings.

The rest of this paper is organized as follows. Section 2 briefly reviews top-k queries over uncertain databases and related works about reverse query processing. Section 3 formally defines our problem of probabilistic reverse top-k query in both monochromatic and bichromatic cases. Section 4 studies in detail the partitions of the solution space of MPRT queries. Section 5 develops the general processing framework, the pruning heuristics, and query procedures for BPRT queries. Section 6 evaluates the performance of our approaches. Finally, Section 7 concludes this paper.

2. Related work

As probabilistic reverse top-k queries are inherently related to probabilistic top-k queries and reverse query processing, some representative works are summarized here.

2.1. Uncertain top-k query processing

While research works on definite top-k queries are mostly based on some deterministic scoring functions, the existence of probability dimension in an uncertain database makes evaluation of uncertain top-k queries very complicated since the top-k query result set depends not only on the aggregated score of candidate objects but also their probabilities [9]. Different combination of the two factors may generate various uncertain top-k query semantics, among which the most popular queries include U-Top$k$ [36,35], U-kRanks [36,30], PT-k [21,22,20], Expected Rank [7,23], E-Score Rank [7,23], Global-Top$k$ [50,51], c-typical-Top$k$ [17], PTD query [31,28], PTkS [32], UTR query [45,46], and PTI query [44] etc.

Soliman et al. [36] proposed firstly the uncertain top-k query semantics, namely U-Top$k$ and U-kRanks. However, on account of lacking of pruning heuristics with a possible world space which grows exponentially, the algorithms proposed are inefficient. Hua et al. [22] presented firstly a probabilistic threshold top-k (PT-k) query which returns those tuples whose top-k probabilities over all possible worlds are no less than a user-specified probability threshold q for improving performance without unfolding the whole possible world space. Zhang et al. [51] put forward a Global-Top$k$ query which reports k highest-ranked tuples based on their top-k probabilities. Particularly, the dynamic programming was applied by the authors for computing the top-k probability of an uncertain object. Cormode et al. [7] proposed Expected Rank and E-Score Rank query semantics for top-k query over uncertain databases. Xiao et al. [45,46,44] firstly proposed an uncertain top-k range (UTR) query which retrieves l uncertain tuples that meet a score range constraint and have the maximum top-k probabilities but no less than a user-defined probability threshold, and the authors given a method of parallel optimization

based on based on multicore architectures. In addition, Xiao et al. proposed a probabilistic top-l influential (PTI) query to identify the l most favorite objects [44].

These query semantics aforementioned are all based on tuple-level uncertainty. For attribute-level uncertainty, Lian and Chen [31] presented the probabilistic top-k dominating (PTD) query, which retrieves k uncertain objects that are expected to dynamically dominate the largest number of objects. In [32], the authors proposed the probabilistic top-k star (PTkS) query, which aims to return k uncertain objects that are "closest" to a static/dynamic query point, considering both distance and probability aspects. Several effective pruning heuristics were also put forward for reducing the query search space in the two papers. In addition, Tao et al. [37] studied the uncertain range query for an arbitrary probability distribution function in a multi-dimensional data space. Li et al. [26] proposed a unified method to ranking the top-k query in probabilistic databases by regarding it as a multi-criteria optimization problem. Yiu et al. [47] studied the probabilistic spatial range queries with R-tree in multi-dimensional spaces for uncertain databases. Tseng et al. [38] studied high utility itemsets (HUIs) mining, which refers to discovering all itemsets having a utility meeting a user-specified minimum utility threshold. In this paper, we only consider tuple-level uncertainty with all objects mutually independent. In practice, almost all tuples are mutually independent in the context of tuple-level uncertainty.

We can see from these query semantics that existing works study uncertain top-k queries from the perspective of users that seek objects matching their preferences. To the best of our knowledge, there are few techniques refereeing to reverse top-k queries over uncertain databases from the perspective of manufacturers.

2.2. Reverse query processing

A variety of reverse query types, which takes a query data point as input and aims to find the queries that have this query point in their answer set have been studied. Lian and Chen [29] proposed firstly the probabilistic reverse skyline queries. Gao et al. [14,15] studied skyline queries and k-skyband queries and proposed efficient reverse skyline and reverse k-skyband query algorithms. Cheema et al. [3] formalized the problem of probabilistic reverse nearest neighbors based on the possible worlds semantics for the first time. However, due to the intrinsic difference of semantics between skyline/RNN queries and top-k queries, they cannot be used to directly handle the probabilistic reverse top-k queries.

Reverse top-k queries [39,42,40,41,43] have been proposed for estimating the impact of a potential product in the market, based on the number of customers that regard this product as one of their top-k products based on their preferences. Recently, various applications of reverse top-k queries have appeared, including identifying the most influential products [42, 25], and monitoring the popularity of locations based on user mobility [41]. Ge et al. [16] presented a unified framework for all top-k computation over traditional certain databases, such as top-k, reverse top-k, and top-m influential queries. Yu et al. [48] studied reverse top-k search using random walk with restart. However, existing works only studied reverse top-k in precise databases. Jin et al. [24] proposed firstly the probabilistic reverse top-k queries upon uncertain data. However, in this related work, each object in the uncertain database is described as a probabilistic distribution function (i.e., the continuous situation) rather than a vector (i.e., the discrete situation) which is more useful in real-life applications to describe the attributes of objects and the preferences of users. The proposed algorithm in this work cannot handle the discrete case directly.

To the best of our knowledge, this is the first work to study the probabilistic reverse top-k query on uncertain data for the discrete situation, considering both monochromatic and bichromatic cases.

3. Problem statement

In this section, we firstly introduce the basics with respect to an uncertain data model and its corresponding possible world semantics. After that, we proceed to define our problems.

3.1. Preliminaries

Consider a d-dimensional uncertain database $\mathcal{D}$ containing $|\mathcal{D}|$ uncertain objects. Each uncertain object $o \in \mathcal{D}$ can be represented by a quality vector $(o[1], o[2], \ldots, o[d])$, where $o[i]$ denotes the evaluation value of o on the ith quality attribute. $Pr(o)$ denotes the appearance probability that object o exists in $\mathcal{D}$. In addition, a set of user preferences $\mathcal{U}$ contains $|\mathcal{U}|$ users is taken into account. Each user $\mu \in \mathcal{U}$ can be formulated by a preference weighting vector $(\mu[1], \mu[2], \ldots, \mu[d])$, where $\mu[j]$ denotes the concerning degree of user μ on the jth quality attribute. Without loss of generality, we assume that the values of $o[i]$ and $\mu[j]$ are numerical non-negative scores and are normalized in $[0, 1]$, and $\sum_{j=1}^{d} \mu[j] = 1$. Besides, smaller score values are preferable. The aggregated score $AS_\mu(o)$ for o with respect to μ is defined as a weighted sum of the individual scores, i.e.,

$$AS_\mu(o) = \sum_{j=1}^{d} o[j] \cdot \mu[j]. \tag{1}$$

There exist many works on modeling uncertain data. One of the most popular is the model based on possible world semantics [9,8,13], where an uncertain database is regarded as a set of possible world instances associated with their

Table 3
An example of possible worlds for the uncertain product data described in Table 1.

Possible World ($\mathcal{W}$)	$Pr(\mathcal{W})$
$\mathcal{W}_1 = \{a, b, c\}$	$0.6 \times 0.7 \times 0.5 = 0.21$
$\mathcal{W}_2 = \{a, b\}$	$0.6 \times 0.7 \times 0.5 = 0.21$
$\mathcal{W}_3 = \{a, c\}$	$0.6 \times 0.3 \times 0.5 = 0.09$
$\mathcal{W}_4 = \{b, c\}$	$0.4 \times 0.7 \times 0.5 = 0.14$
$\mathcal{W}_5 = \{a\}$	$0.6 \times 0.3 \times 0.5 = 0.09$
$\mathcal{W}_6 = \{b\}$	$0.4 \times 0.7 \times 0.5 = 0.14$
$\mathcal{W}_7 = \{c\}$	$0.4 \times 0.3 \times 0.5 = 0.06$
$\mathcal{W}_8 = \{\emptyset\}$	$0.4 \times 0.2 \times 0.5 = 0.04$

Table 4
An example of PTT query and BPRT query for the uncertain product data set and user preference set described in Tables 1 and 2.

$Pr^2_\mu(\cdot)$	a	b	c	PTT_2
Alice	0.6	0.49	0.5	$\{a, c\}$
Bob	0.39	0.7	0.5	$\{b, c\}$
Chris	0.6	0.7	0.29	$\{b, a\}$
$bPRT_2(q)$	{Alice, Chris}	{Bob, Chris}	{Alice, Bob}	–

probabilities. Each possible world $\mathcal{W}$ is a subset of uncertain data objects, and the set of all worlds is denoted by the possible world space Ω. The probability of each world is computed as the joint probability of the existence of the world's objects and the absence of all other data objects. Since all objects are mutually independent, we can obtain:

$$Pr(\mathcal{W}) = \prod_{t \in \mathcal{W}} Pr(t) \cdot \prod_{t \notin \mathcal{W}} (1 - Pr(t)), \tag{2}$$

where $\sum_{\mathcal{W} \in \Omega} Pr(\mathcal{W}) = 1$. Table 3 shows an example about the possible worlds for the uncertain database in two dimensional space based on the house property database described in Table 1.

3.2. Probabilistic top-k queries

The answer set of a definite top-k query is a ranked list of k objects with k smallest aggregate scoring values. We firstly give the definition of the traditional definite top-k query.

Definition 1 *(Top-k query).* Given $\mathcal{D}$, μ, and a positive integer k, the top-k query of μ on $\mathcal{D}$, denoted by TOP_k, is a set of objects such that $TOP_k \subseteq \mathcal{D}$, $|TOP_k| = k$, and for $\forall o, t$, $o \in TOP_k$, $t \in \mathcal{D} \backslash TOP_k$, $AS_\mu(o) < AS_\mu(t)$.

Consider for example possible world $\mathcal{W}_1 = \{a, b, c\}$ for Chris's preferences, the TOP_2 of Chris is $\{a, b\}$ on the basis of $AS_{Chris}(\cdot)$. A delicate situation arises when two (or more) objects share the same aggregate scoring value for the kth position. In this case, for simplicity reasons, we assume that it reports all of them as top-k results. For instance, $AS_{Chris}(a) = AS_{Chris}(b) = 0.45$, we report $\{a, b\}$ as the top-2 results of Chris.

Definition 2 *(Top-k probability [22]).* Given $\mathcal{D}$, $\mathcal{U}$, and k, the top-k probability of any object $o \in \mathcal{D}$ w.r.t. $\mu \in \mathcal{U}$, denoted by $Pr^k_\mu(o)$, can be defined as the summation of the probabilities of all possible worlds whose top-k answer set contains o, i.e.,

$$Pr^k_\mu(o) = \sum_{\mathcal{W} \in \Omega,\, o \in TOP_k} Pr(\mathcal{W}). \tag{3}$$

Table 4 illustrates the top-2 probabilities for each user preference based on the house property database depicted in Table 1. For instance, the top-2 probabilities of a, b, and c based on Alice's preferences are 0.6, 0.49, and 0.5, respectively.

In [51], the authors provided a method of computing the top-k probabilities of an object in an uncertain database, that is, for an uncertain data set $\mathcal{D}$ with cardinality $|\mathcal{D}|$, a positive integer k, an aggregate scoring function $AS_\mu(\cdot)$, and the data set is sorted in the ascending order of the $AS_\mu(\cdot)$, denoted by $\mathcal{D} = \{o_1, o_2, \ldots, o_{|\mathcal{D}|}\}$. The top-$k$ probability of uncertain object o_i has the following recursion:

$$Pr^k_\mu(o_i) = \begin{cases} Pr(o_i), & 1 \le i \le k \\ \left\{ Pr^k_\mu(o_{i-1}) \cdot \frac{1 - Pr(o_{i-1})}{Pr(o_{i-1})} + Pr^{k-1}_\mu(o_{i-1}) \right\} \cdot Pr(o_i), & i > k. \end{cases} \tag{4}$$

Table 5
Frequently used symbols and their descriptions.

Symbol	Description
$\mathcal{D}$	An uncertain database
$\|\mathcal{D}\|$	The cardinality of set $\mathcal{D}$
o	An uncertain object in $\mathcal{D}$
$\mathcal{U}$	The set of user preference weighting vectors
μ	A user preference weighting vector in $\mathcal{U}$
$Pr(o)$	Object o existing probability in $\mathcal{D}$
d	The size of dimensionality
$\mathcal{W}$	A possible world
$Pr(\mathcal{W})$	The probability of $\mathcal{W}$
Ω	Possible world space
$AS_\mu(\cdot)$	The aggregate scoring function
k	A positive integer
α	The user-specified probability threshold
$Pr^k_\mu(o)$	The top-k probability of an uncertain object o
TOP_k	A deterministic top-k query
PTT_k	A probabilistic threshold top-k query
$mPRT_k(q)$	A monochromatic probabilistic reverse top-k query
$bPRT_k(q)$	A bichromatic probabilistic reverse top-k query

Use Eq. (4), we can obtain the same results concerning top-2 probabilities of the three uncertain house objects.

In real-life applications, the top-k probabilities of some objects are very small, leading to not only a low value of practical application but also a significant waste of computing resources. Therefore, researchers usually focus on those tuples whose top-k probabilities are greater than or equal to a user-specified probability threshold.

Definition 3 *(Probabilistic threshold top-k (PTT) query).* Given $\mathcal{D}$, μ, k, and a probability threshold $\alpha \in (0, 1]$, the PTT query of μ on $\mathcal{D}$, denoted by PTT_k, is a set of uncertain objects such that $PTT_k \subseteq \mathcal{D}$, and for $\forall o$, t, $o \in PTT_k$, $t \in \mathcal{D} \backslash PTT_k$, $Pr^k_\mu(o) \geq \alpha > Pr^k_\mu(t)$.

The uncertain top-2 houses are depicted in bold in Table 4 for a given probability threshold 0.5. For instance, the probabilistic threshold top-2 query answer set of Alice is $\{a, c\}$. That is, Alice is more likely to purchase Houses a and c.

3.3. Probabilistic reverse top-k queries

In this subsection, we formally define the monochromatic and the bichromatic probabilistic reverse top-k queries.

Definition 4 *(Monochromatic probabilistic reverse top-k (MPRT) query).* Given $\mathcal{D}$, k, α, and an uncertain query object q in $\mathcal{D}$, the MPRT query for q, denoted by $mPRT_k(q)$, is the set $\mathcal{U}$ of d-dimensional vectors, for which $\forall \mu \in \mathcal{U}, q \in PTT_k$.

For the MPRT query, it aims to describe the partitions of the solution space that satisfy the query and estimate the impact of a product in the market based on returned user community when no user preferences are given. In the next section, we will show that the solution space of the MPRT query is a union of some high-dimensional cells defined by a collection of hyperplanes, and in each cell, the ranking order of objects is the same.

Definition 5 *(Bichromatic probabilistic reverse top-k (BPRT) query).* Given $\mathcal{D}$, $\mathcal{U}$, k, α, and an uncertain query object q in $\mathcal{D}$, the BPRT query for q, denoted by $bPRT_k(q)$, consists of the users in $\mathcal{U}$ whose PTT queries contain q. In other words, $bPRT_k(q) \subseteq \mathcal{U}$, and for $\forall \mu \in bPRT_k(q), q \in PTT_k$.

In the real world, given a set of user preferences, BPRT queries for a potential product have even wider applicability since they identify those users that are interested in the potential product. Consider for example the uncertain house property data set depicted in Table 1. For an uncertain query object $q = a$, its bichromatic probabilistic reverse top-2 result set is {Alice, Chris}. Likewise, the answer sets for b and c are {Bob, Chris} and {Alice, Bob}, respectively, as illustrated in Table 4.

In brief, the bichromatic version of the probabilistic reverse top-k query returns a finite number of user preferences, while the monochromatic version determines the partitions of the solution space that satisfy the query and estimates the impact of the query product based on the returned user community.

The frequently used symbols and their descriptions in this paper are summarized in Table 5.

4. Monochromatic probabilistic reverse top-*k* queries

In this section, we firstly study the partitions of solution space of MPRT queries. Then, we provide in theory mathematical model for solving MPRT queries in an arbitrary dimensional data space.

Table 6
The ranking order of three objects and their top-k ($k = 1, 2$) probabilities for 2-dimensional space.

The order of $AS_\mu(\cdot)$	μ_1	μ_2	$Pr^1_\mu(a)$	$Pr^2_\mu(a)$	$Pr^1_\mu(b)$	$Pr^2_\mu(b)$	$Pr^1_\mu(c)$	$Pr^2_\mu(c)$
$AS_\mu(a) < AS_\mu(b) < AS_\mu(c)$	(2/5, 1/2)	(1/2, 3/5)	0.6	0.6	0.28	0.7	0.06	0.29
$AS_\mu(a) < AS_\mu(c) < AS_\mu(b)$	[0, 2/5)	(3/5, 1]	0.6	0.6	0.14	0.49	0.2	0.5
$AS_\mu(b) < AS_\mu(a) < AS_\mu(c)$	(1/2, 4/7)	(3/7, 1/2)	0.18	0.6	0.7	0.7	0.06	0.29
$AS_\mu(b) < AS_\mu(c) < AS_\mu(a)$	(4/7, 1]	[0, 3/7)	0.09	0.39	0.7	0.7	0.15	0.5
$AS_\mu(c) < AS_\mu(a) < AS_\mu(b)$	$\emptyset$	$\emptyset$	–	–	–	–	–	–
$AS_\mu(c) < AS_\mu(b) < AS_\mu(a)$	$\emptyset$	$\emptyset$	–	–	–	–	–	–

4.1. Interpretation of solution space

We firstly introduce a pruning heuristic before interpreting of solution space. In a PTT query, the top-k probability of an uncertain object is at most its existence probability [22,45]. Then, we have the following Lemma 1 which can avoid checking some objects that cannot satisfy the probability threshold α.

Lemma 1 *([22,45]). Given α and $o \in \mathcal{D}$. If $Pr(o) < \alpha$, then for $\forall\mu$, $o \notin PTT_k$.*

For a given query object q, we firstly estimate that whether or not $Pr(q)$ is greater than or equal to the probability threshold α when processing MPRT queries. If $Pr(q) < \alpha$, according to Lemma 1, for $\forall\mu$, $q \notin PTT_k$, thereby $\mu \notin mPRT_k(q)$.

In a d-dimensional space, given $\mathcal{D}$, $q \in \mathcal{D}$, and a PTT query, an MPRT query of q retrieves all users μ, for which $q \in PTT_k$. Assume that $\mathcal{U}$ denotes the set of all valid assignments of μ. Since $\sum_{j=1}^{d} \mu[j] = 1$ and $\mu[j] \in [0, 1]$, essentially, $\mathcal{U}$ is a part of the space defined by the hyperplanes $\sum_{j=1}^{d} \mu[j] = 1$ and $\mu[j] = 0$ ($j = 1, 2, \cdots, d$). All user preferences that lie in $\mathcal{U}$ are all the monochromatic probabilistic reverse top-k query results. It is impossible to enumerate all possible assignments of $\mu \in \mathcal{U}$, since the number of possible vectors μ is infinite. On the other hand, we can determine the boundaries of $\mathcal{U}$. Generally, the solution space $\mathcal{U}$ can be split into a finite set of independent partitions $\mathcal{U}_j$ ($\cup\mathcal{U}_j = \mathcal{U}$, $\cap\mathcal{U}_j = \emptyset$) based on the rank of q in possible worlds. For each $\mu \in \mathcal{U}_j$, q has the same ranking position. Then, the answer set of the MPRT query is a set of partitions $\mathcal{U}_j$ of the solution space $\mathcal{U}$:

$$mPRT_k(q) = \{\mathcal{U}_j | \mu_i \in \mathcal{U}_j, q \in PTT_k\}. \tag{5}$$

In other words, the solution space $\mathcal{U}$ of the MPRT query is a union of some high-dimensional cells defined by a collection of hyperplanes, and in each cell, the ranking order of objects is same. Particularly, it is a union of some planes defined by a collection of line segments in a 2-dimensional space.

Definition 6 *(Impact of query object).* In order to evaluate the impact of the query object q being ranked in the kth in the market, we give the definition of IF_q^k, it denotes the proportion of the solution space of MPRT query in the total space:

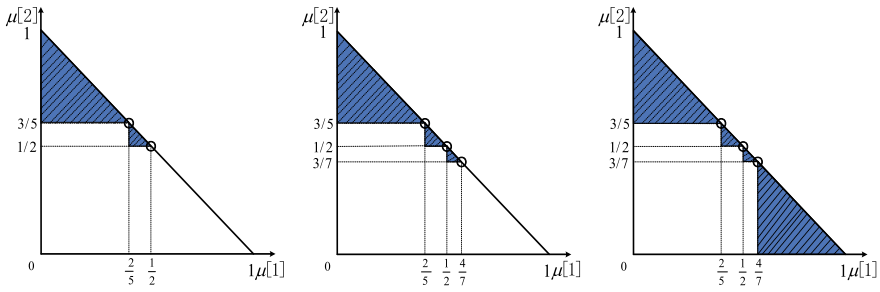
$$IF_q^k = \frac{\mathscr{S}}{\mathscr{V}} \times 100\%, \tag{6}$$

where $\mathscr{S}$ is the measure of the solution space of MPRT query of q and $\mathscr{V}$ is the measure of the space defined by the hyperplanes $\sum_{j=1}^{d} \mu[j] = 1$ and $\mu[j] = 0$ ($j = 1, 2, \cdots, d$) in the d-dimensional space.

Please note that the measure of the space can be computed by an (multiple) integration for the spatial graphic enclosed by the boundaries of the space.

An illustrated example for 2-dimensional space. In order to determine the boundaries of the partition $\mathcal{U}_j$ of the solution space $\mathcal{U}$, we consider for example the uncertain house property data set depicted in Table 1. For $\forall\mu = (\mu[1], \mu[2])$, the aggregate scoring values of a, b, and c are $AS_\mu(a) = 0.8\mu[1] + 0.1\mu[2]$, $AS_\mu(b) = 0.2\mu[1] + 0.7\mu[2]$, and $AS_\mu(c) = 0.5\mu[1] + 0.5\mu[2]$, respectively. Now, given a potential house a and a probability threshold (trustability) $\alpha = 0.5$, what are the user preferences for which a is in the probabilistic threshold top-1 results?

The rank of a is related to the relative order of between a and other data objects in $\mathcal{D}$. Generally, the number of possible orders of these objects in $\mathcal{D}$ is its full-permutation ($P_{|\mathcal{D}|}^{|\mathcal{D}|}$). For each permutation, there exists a ranking order of objects and accordingly a possible solution space. There are six (i.e., $P_3^3 = 6$) possible ranking orders for the three objects as illustrated in Table 6. For the situations of $AS_\mu(b) < AS_\mu(a) < AS_\mu(c)$ and $AS_\mu(b) < AS_\mu(c) < AS_\mu(a)$, the top-1 probabilities of a are $Pr^1_\mu(a) = 0.18 < \alpha$ and $Pr^1_\mu(a) = 0.09 < \alpha$, respectively. On the other hand, there are no solution sets for the cases of $AS_\mu(c) < AS_\mu(a) < AS_\mu(b)$ and $AS_\mu(c) < AS_\mu(b) < AS_\mu(a)$, respectively. In the four cases, $a \notin PTT_1$, thereby $mPRT_1(a) = \emptyset$. Only when $AS_\mu(a) < AS_\mu(b) < AS_\mu(c)$ or $AS_\mu(a) < AS_\mu(c) < AS_\mu(b)$, that is, a is ranked at the first position in possible worlds, $Pr^1_\mu(a) = 0.6 > \alpha$, and $a \in PTT_1$ in the two cases. Solving the inequations under constraints $\mu[1] + \mu[2] = 1$, and $\mu[1]$, $\mu[2] \in [0, 1]$, respectively, we obtain $\mathcal{U}_{11} = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in (2/5, 1/2), \mu_j[2] \in (1/2, 3/5)\}$



(a) $mPRT_1(a)$. (b) $mPRT_2(a)$. (c) $mPRT_3(a)$.

Fig. 1. An example of the solution space of MPRT queries in 2-dimensional space.

and $\mathcal{U}_{12} = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in [0, 2/5), \mu_j[2] \in (3/5, 1]\}$, respectively, as illustrated in blue parts of Fig. 1(a). They are the solution space of the monochromatic probabilistic reverse top-1 query of a, which is a part of space defined by line segment $\mu[1] + \mu[2] = 1$, $\mu[1] = 0$, and $\mu[2] = 0$. Furthermore, the impact of a being ranked in 1st in the market can be estimated as:

$$\begin{aligned} IF_a^1 &= \frac{\mathscr{S}_1}{\mathscr{V}} = \frac{\mathscr{S}_{11} + \mathscr{S}_{12}}{\mathscr{V}} \\ &= \frac{\frac{1}{2} \cdot (\frac{1}{2} - \frac{2}{5}) \cdot (\frac{3}{5} - \frac{1}{2}) + \frac{1}{2} \cdot (\frac{2}{5} - 0) \cdot (1 - \frac{3}{5})}{\frac{1}{2} \cdot 1 \cdot 1}, \\ &= 17\% \end{aligned} \tag{7}$$

where $\mathscr{S}_1$, $\mathscr{S}_{11}$, and $\mathscr{S}_{12}$ are the area of $\mathcal{U}_1$, $\mathcal{U}_{11}$, and $\mathcal{U}_{12}$, respectively, $\mathscr{V}$ is the area of $\mathcal{U}$: $\{(\mu_j[1], \mu_j[2]) | \mu_j[1] + \mu_j[2] = 1, \mu_j[1] \in [0, 1], \mu_j[2] \in [0, 1]\}$.

Similarly, the monochromatic probabilistic reverse top-2 result set of a is $\mathcal{U}_2 = \cup_{i=1}^{3} \mathcal{U}_{2i}$, where $\mathcal{U}_{21} = \mathcal{U}_{11}$, $\mathcal{U}_{22} = \mathcal{U}_{12}$, and $\mathcal{U}_{23} = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in (1/2, 4/7), \mu_j[2] \in (3/7, 1/2)\}$, as described in blue parts of Fig. 1(b). Likewise, $mPRT_3(a) = \mathcal{U}_3 = \cup_{i=1}^{4} \mathcal{U}_{3i}$, where $\mathcal{U}_{3i} = \mathcal{U}_{2i}$ $(i = 1, 2, 3)$, and $\mathcal{U}_{34} = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in (4/7, 1], \mu_j[2] \in [0, 3/7)\}$, as illustrated in blue parts of Fig. 1(c). Furthermore, the impacts of a being ranked in the 2nd and the 3rd in the market are

$$IF_a^2 = \frac{\mathscr{S}_2}{\mathscr{V}} = 17.51\%, \tag{8}$$

and

$$IF_a^3 = \frac{\mathscr{S}_3}{\mathscr{V}} = 35.88\%, \tag{9}$$

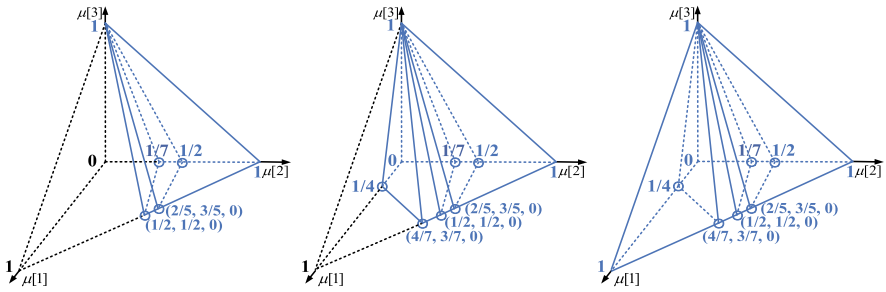
respectively, where $\mathscr{S}_2$ and $\mathscr{S}_3$ are the areas of $\mathcal{U}_2$ and $\mathcal{U}_3$, respectively, $\mathscr{V}$ is the area of $\mathcal{U}$: $\{(\mu_j[1], \mu_j[2]) | \mu_j[1] + \mu_j[2] = 1, \mu_j[1] \in [0, 1], \mu_j[2] \in [0, 1]\}$.

In the following, we will give an brief introduction for determining the solution space of the MPRT query in higher dimensional data space.

4.2. Higher dimensional data

In higher dimensions ($d > 2$), an MPRT query returns a union of some high-dimensional cells defined by a collection of hyperplanes.

An illustrated example for 3-dimensional space. Let us consider the data set as described in Table 1, additional attributes of surrounding environment 0.4, 0.3, and 0.5 are considered for Houses a, b, and c, respectively. Similar to the 2-dimensional case, the boundaries of the cells are defined by the weighting vectors. The rank of a depends on the relative order between a and the other two objects. Only when $AS_\mu(a) < AS_\mu(b) < AS_\mu(c)$ or $AS_\mu(a) < AS_\mu(c) < AS_\mu(b)$, $Pr_\mu^1(a) = 0.6 > \alpha$, and $a \in PTT_1$ in the two cases. The solution space $\mathcal{U}_1$ contains two 3-dimensional cells (in fact, a triangular pyramid $\mathcal{U}_{12}$ and a rectangular pyramid $\mathcal{U}_{11}$, as illustrated with blue lines in Fig. 2(a)): $\mathcal{U}_1 = \mathcal{U}_{11} \cup \mathcal{U}_{12}$, where $\mathcal{U}_{11} = \{(\mu_j[1], \mu_j[2], \mu_j[3]) | \mu_j[3] = 1 - \mu_j[1] - \mu_j[2], (\mu_j[1], \mu_j[2]) \in D_1\}$, $D_1 = D_{11} \cup D_{12}$, $D_{11} = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in [0, 2/5], \mu_j[2] \in (\frac{5\mu_j[1]+1}{7}, \frac{\mu_j[1]+2}{4})\}$ and $D_{12} = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in [2/5, 1/2), \mu_j[2] \in (\frac{5\mu_j[1]+1}{7}, 1 - \mu_j[1]]\}$; $\mathcal{U}_{12} = \{(\mu_j[1], \mu_j[2], \mu_j[3]) | \mu_j[3] = 1 - \mu_j[1] - \mu_j[2], (\mu_j[1], \mu_j[2]) \in D_2\}$, $D_2 = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in [0, 2/5), \mu_j[2] \in (\frac{\mu_j[1]+2}{4}, 1 - \mu_j[1]]\}$. Furthermore, the impact of



(a) $mPRT_1(a)$. (b) $mPRT_2(a)$. (c) $mPRT_3(a)$.

Fig. 2. An example of the solution space of MPRT queries in 3-dimensional space.

a being ranked in the 1st in the market can be estimated as:

$$\begin{aligned} IF_a^1 &= \frac{\mathscr{S}_1}{\mathscr{V}} \\ &= \frac{\frac{1}{3}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(1-\frac{1}{7})\cdot 1}{\frac{1}{3}\cdot\frac{1}{2}\cdot 1\cdot 1\cdot 1}, \\ &= 42.86\% \end{aligned} \tag{10}$$

where, $\mathscr{S}_1$ is the volume of $\mathcal{U}_1$ and $\mathscr{V}$ is the volume of $\mathcal{U} = \{(\mu_j[1], \mu_j[2], \mu_j[3]) | \sum_{i=1}^{3} \mu_j[i] = 1, \mu_j[i] \in [0,1]\}$.

Similarly, the monochromatic probabilistic reverse top-2 result set of a is $\mathcal{U}_2 = \cup_{i=1}^{3}\mathcal{U}_{2i}$ (as described with blue lines in Fig. 2(b), it is a rectangular pyramid consisted of a triangular pyramid $\mathcal{U}_{22}$, a rectangular pyramid $\mathcal{U}_{21}$, and a pentagonal pyramid $\mathcal{U}_{23}$), where $\mathcal{U}_{21} = \mathcal{U}_{11}$, $\mathcal{U}_{22} = \mathcal{U}_{12}$, and $\mathcal{U}_{23} = \{(\mu_j[1], \mu_j[2], \mu_j[3]) | \mu_j[3] = 1 - \mu_j[1] - \mu_j[2], (\mu_j[1], \mu_j[2]) \in D_3\}$, $D_3 = \cup_{i=1}^{3} D_{3i}$, $D_{31} = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in [0, 1/4], \mu_j[2] \in [0, \frac{5\mu_j[1]+1}{7})\}$, $D_{32} = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in [1/4, 1/2], \mu_j[2] \in (\frac{4\mu_j[1]+1}{3}, \frac{5\mu_j[1]+1}{7})\}$, and $D_{33} = \{(\mu_j[1], \mu_j[2]) | \mu_j[1] \in [1/2, 4/7), \mu_j[2] \in (\frac{4\mu_j[1]+1}{3}, 1 - \mu_j[1]]\}$. Likewise, $mPRT_3(a) = \mathcal{U}_3 = \cup_{i=1}^{4}\mathcal{U}_{3i}$ (as described with blue lines in Fig. 2(c), it is a rectangular pyramid consisted of two triangular pyramids, namely $\mathcal{U}_{32}$ and $\mathcal{U}_{34}$, a rectangular pyramid $\mathcal{U}_{31}$, and a pentagonal pyramid $\mathcal{U}_{33}$), where $\mathcal{U}_{3i} = \mathcal{U}_{2i}$ $(i = 1, 2, 3)$ and $\mathcal{U}_{34} = \{(\mu_j[1], \mu_j[2], \mu_j[3]) | \mu_j[3] = 1 - \mu_j[1] - \mu_j[2], (\mu_j[1], \mu_j[2]) \in D_4\}$, $D_4 = \{(\mu_j[1], \mu_j[2]) | \mu_j[2] \in [0, 3/7), \mu_j[2] \in (\frac{3\mu_j[2]+1}{4}, 1 - \mu_j[2]]\}$. Furthermore, the impacts of a being ranked in the 2nd and the 3rd in the market are

$$\begin{aligned} IF_a^2 &= \frac{\mathscr{S}_2}{\mathscr{V}} \\ &= \frac{\frac{1}{3}\cdot[\frac{1}{2}\cdot 1\cdot 1 - \frac{1}{2}\cdot(1-\frac{1}{4})\cdot\frac{3}{7}]\cdot 1}{\frac{1}{3}\cdot\frac{1}{2}\cdot 1\cdot 1\cdot 1}, \\ &= 67.86\% \end{aligned} \tag{11}$$

and

$$IF_a^3 = \frac{\mathscr{S}_3}{\mathscr{V}} \approx 1, \tag{12}$$

respectively, where $\mathscr{S}_2$, $\mathscr{S}_3$ are the volumes of $\mathcal{U}_2$ and $\mathcal{U}_3$, respectively, $\mathscr{V}$ is the volume of triangular pyramid $\mathcal{U} = \{(\mu_j[1], \mu_j[2], \mu_j[3]) | \sum_{i=1}^{3} \mu_j[i] = 1, \mu_j[i] \in [0,1]\}$. We can apply multiple integral to estimate the measure of the solution space for higher dimensional space.

The methods for solving MPRT queries in 2/3-dimensional space can be extended for higher dimensional data. For a given query object q, if $Pr(q) \geq \alpha$, we process q as follows. For $\forall\mu = (\mu[1], \mu[2], \ldots, \mu[d])$, we compute the aggregate scoring value $AS_\mu(o)$ for each object o in $\mathcal{D}$ based on μ. In order to ensure $q \in PTT_k$, $Pr_\mu^k(q)$ must be greater than or equal to probability threshold α. Similar to 2/3-dimensional case, we heuristically compare $AS_\mu(q)$ to $AS_\mu(\cdot)$ of other objects, so that we can determine the rank of q in these possible worlds, thereby computing $Pr_\mu^k(q)$. The problem can be transformed into a solving for a linear matrix inequality under constraints $\sum_{i=1}^{d} \mu[i] = 1$ and $\mu[i] \in [0,1]$, $i = 1, 2, \cdots, d$, i.e.,

$$\begin{cases} A\mu^T < 0 \\ \sum_{i=1}^{d} \mu[i] = 1 \\ \mu[i] \in [0,1], i = 1, 2, \cdots, d, \end{cases} \tag{13}$$

Algorithm 1 BPRT_query processing framework.

Input:
$\mathcal{D}$, $\mathcal{U}$, $q \in \mathcal{D}$, α, and k
Output:
$bPRT_k(q)$
1: construct a multidimensional spatial index structure, i.e., $\mathcal{I}_\mathcal{D}$ and $\mathcal{I}_\mathcal{U}$, for $\mathcal{D}$ and $\mathcal{U}$, respectively; //indexing phase
2: perform pruning heuristics over the indexes; //pruning phase
3: refine the retrieved candidates and return the answer set. //refinement phase

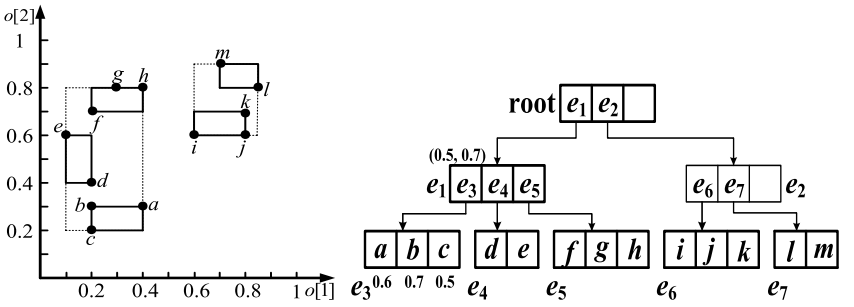


Fig. 3. An example of the probabilistic R-tree.

where, μ^T is the transposition of μ, $A = (o_{i_1} - o_{j_1}, o_{i_2} - o_{j_2}, \cdots, o_{i_{|\mathcal{D}|}} - o_{j_{|\mathcal{D}|}},)$, $i_1\ i_2\ \cdots\ i_{|\mathcal{D}|}$ and $j_1\ j_2\ \cdots\ j_{|\mathcal{D}|}$ are one of permutations of $1\ 2\ \cdots\ |\mathcal{D}|$, $i_k \neq j_k$, $i, j = 1, 2, \cdots, |\mathcal{D}|$. As previously mentioned, the number of possible orders of these objects in $\mathcal{D}$ is its full-permutation $P_{|\mathcal{D}|}^{|\mathcal{D}|}$, each A corresponds to a permutation. Solving each linear matrix inequality under constraints $\sum_{i=1}^{k} \mu[i] = 1$, $\mu[i] \in [0, 1]$, and $Pr_\mu^k(q) \geq \alpha$, we can obtain a possible solution domain of $mPRT_k(q)$ defined by a set of hyperplanes. All these possible solution domains form the solution space of the MPRT query of q. It's worth noting that in a 2-dimensional data space, it is a union of some planes defined by a collection of line segments while a union of some higher dimensional cells defined by a set of hyperplanes in a d-dimensional data space ($d \geq 3$).

At present, the solving with respect to a linear matrix inequality focuses on mainly its feasibility problem. We can easily find a feasible solution μ but a solution space for the linear matrix inequality by MatLab LMI or YaLMIP toolkits. Guo et al. [19] proposed a method for solving a linear matrix inequality. However, the method is only effective for a symmetric coefficient matrix A. To the best of our knowledge, there is no an effective solving method for a generalized coefficient matrix. The details of such a generalization are very interesting and challenging, we are going to study them further in our future work.

5. Bichromatic probabilistic reverse top-*k* queries

In this section, we mainly study BPRT queries for a potential product which is more useful in the real world applications. We firstly give the BPRT query processing framework in Section 5.1. Then several pruning heuristics are proposed in Section 5.2. In Section 5.3, we give the detailed BPRT query processing algorithm.

5.1. The general framework

Different from MPRT queries, the BPRT problem involves two data sets, i.e., $\mathcal{D}$ and $\mathcal{U}$, where $\mathcal{D}$ represents the uncertain data objects and $\mathcal{U}$ denotes the user preferences. The goal of this work is to find all preferences $\mu \in \mathcal{U}$ such that the $q \in PTT_k$ for a user-specified probability threshold α. Such a query, if computed in a straightforward manner without any pruning heuristic (***naïve algorithm, NA, for short***), requires evaluating a PTT query for each μ in $\mathcal{U}$, which is prohibitively expensive and does not scale even for moderate data sets, let alone the big $\mathcal{U}$ and $\mathcal{D}$. Especially nowadays big data is commonly stored and used in scientific research and business application.

Algorithm 1 illustrates a general framework for answering the BPRT query. It consists of three phases, indexing, pruning, and refinement. In the first indexing phase, given $\mathcal{D}$ and $\mathcal{U}$, we construct a probabilistic R-tree (PR-tree) index $\mathcal{I}_\mathcal{D}$ over $\mathcal{D}$ and an R-tree index $\mathcal{I}_\mathcal{U}$ over $\mathcal{U}$ to facilitate the BPRT query (line 1).

In particularly, each MBR (minimum bounding rectangle) m of R-tree is depicted by the coordinates of its two opposite corners, i.e., the lower-left corner ($m.l$) and the upper-right corner ($m.u$). Note that points $m.l$ and $m.u$ are generally two virtual points, they also can be the data objects in data set. For each intermediate entry e of PR-tree, the following information is stored: a pointer referencing its child entry, an MBR m which includes all uncertain objects in child nodes of e, a presence probability $Pr(m.u)$, which equals to the minimum existence probability of the elements rooted at e, and an occurrence probability $Pr(m.l)$, which equals to the maximum existence probability of the elements rooted at e [10,53,54].

Fig. 3 depicts an example of data sets indexed by a PR-tree. As shown in the figure, the node capacity of the PR-tree equals to 3, and the presence probabilities of three uncertain objects a, b, and c are 0.6, 0.7, and 0.5, respectively. Conse-

quently, their upper level intermediate entry e_3 corresponding to the MBR m of a, b, and c has the existence probabilities $Pr(m.l) = max\{0.6, 0.7, 0.5\} = 0.7$ and $Pr(m.u) = min\{0.6, 0.7, 0.5\} = 0.5$.

As a second step, the pruning phase removes those users that cannot be the results of BPRT queries, using several pruning heuristics without evaluating the corresponding PTT queries (line 2). It can significantly reduce the search space of BPRT queries and thus is efficient for uncertain big data processing. Finally, for each remaining candidate μ that cannot be pruned, the refinement phase computes the top-k probability of query object q based on μ, and evaluates whether the query point belongs to the PTT query for a user-specified probability threshold (line 3). In the following, we mainly focus on the pruning phase and the refinement phase.

5.2. Pruning heuristics

In this section, $\mathcal{D}$ and $\mathcal{U}$ are indexed by a PR-tree and an R-tree, $\mathcal{I}_{\mathcal{D}}$ and $\mathcal{I}_{\mathcal{U}}$, respectively. The aim is to provide effective pruning methods to reduce the search space of BPRT queries.

Vlachou et al. [43] analyzed the effect of a set $\mathcal{V} \subseteq \mathcal{U}$ on the score(s) of a single data object and a set of data points represented by an MBR m, and provided a pruning property based on the score bounds on points and MBRs. However, the pruning heuristics they proposed were only fit for a deterministic database, which cannot be used to directly handle uncertain data. We extend the idea in [43] and provide new strategies for efficiently pruning the user preferences. We firstly give the score bounds on uncertain objects and MBRs, respectively.

Lemma 2 *(Lower and upper bounds of score $AS_\mu(o)$). Given $o \in \mathcal{D}$ and a set of user preferences $\mathcal{V} \subseteq \mathcal{U}$ represented by an MBR $m_{\mathcal{V}}$, the aggregated score $AS_\mu(o)$ of o holds that, for $\forall \mu \in m_{\mathcal{V}}$,*

$$LB\_AS_{m_{\mathcal{V}}}(o) \leq AS_\mu(o) \leq UB\_AS_{m_{\mathcal{V}}}(o), \tag{14}$$

where,

$$LB\_AS_{m_{\mathcal{V}}}(o) = \sum_{j=1}^{d} min_{\forall \mu \in m_{\mathcal{V}}}(\mu[j]) \cdot o[j], \tag{15}$$

$$UB\_AS_{m_{\mathcal{V}}}(o) = \sum_{j=1}^{d} max_{\forall \mu \in m_{\mathcal{V}}}(\mu[j]) \cdot o[j]. \tag{16}$$

Lemma 3 *(Lower and upper bounds of score $AS_\mu(m)$). Given an MBR m which represents a set of objects in $\mathcal{D}$ and a set of user preferences $\mathcal{V} \subseteq \mathcal{U}$ represented by an MBR $m_{\mathcal{V}}$, the aggregated score $AS_\mu(m)$ of m holds that, for $\forall \mu \in m_{\mathcal{V}}$ and $\forall o \in m$,*

$$LB\_AS_{m_{\mathcal{V}}}(m) \leq AS_\mu(o) \leq UB\_AS_{m_{\mathcal{V}}}(m), \tag{17}$$

where,

$$LB\_AS_{m_{\mathcal{V}}}(m) = \sum_{j=1}^{d} min_{\forall \mu \in m_{\mathcal{V}}}(\mu[j]) \cdot m.l[j], \tag{18}$$

$$UB\_AS_{m_{\mathcal{V}}}(m) = \sum_{j=1}^{d} max_{\forall \mu \in m_{\mathcal{V}}}(\mu[j]) \cdot m.u[j]. \tag{19}$$

Note that in the case in which an MBR $m_{\mathcal{V}}$ that encloses the set of user preferences $\mathcal{V}$ is given, the definition of the score bounds $LB\_AS_{m_{\mathcal{V}}}(o)$ and $UB\_AS_{m_{\mathcal{V}}}(o)$ of the aggregated score $AS_{m_{\mathcal{V}}}(o)$ for an object o can be derived by means of the lower-left corner and the upper-right corner of the MBR $m_{\mathcal{V}}$, i.e., $m_{\mathcal{V}}.l$ and $m_{\mathcal{V}}.u$, respectively. Consider for example the real estate deal database depicted in Table 1. The set $\mathcal{V}$ of user preference weighting vectors is enclosed by an MBR $m_{\mathcal{V}}$ defined by the lower-left corner $m_{\mathcal{V}}.l = (0.1, 0.2)$ and the upper-right corner $m_{\mathcal{V}}.u = (0.8, 0.9)$. Then, for the uncertain object $a = (0.8, 0.4)$, the score bounds of a are $LB\_AS_{m_{\mathcal{V}}}(a) = 0.8 \cdot 0.1 + 0.4 \cdot 0.2 = 0.16$ and $UB\_AS_{m_{\mathcal{V}}}(a) = 0.8 \cdot 0.8 + 0.4 \cdot 0.9 = 1$, respectively.

After providing the score bounds for uncertain objects and MBRs, we are now ready to illustrate the intuition of our pruning heuristics. As discussed Lemmas 2 and 3, given a set of preferences $\mathcal{V} \subseteq \mathcal{U}$ represented by an MBR $m_{\mathcal{V}}$, instead of expensively computing the exact scores of uncertain objects and MBRs, we can obtain their score intervals at a lower cost, into which the actual scores fall. For instance, for an object o, $\forall \mu \in m_{\mathcal{V}}$, we have $AS_\mu(o)$ of o within interval $[LB\_AS_{m_{\mathcal{V}}}(o), UB\_AS_{m_{\mathcal{V}}}(o)]$. Similarly, for a set of data objects represented by an MBR m, the aggregated score $AS_\mu(p)$ of each object $p \in m$ belongs to the interval $[LB\_AS_{m_{\mathcal{V}}}(m), UB\_AS_{m_{\mathcal{V}}}(m)]$. Then, we can obtain the following three cases about the aggregated score between the uncertain query point q and a set of objects represented by an MBR m. As illustrated in Fig. 4, where the horizontal axis indicates the uncertain data items (i.e., an uncertain query point q and an MBR m), and the vertical axis represents the aggregated scores of the corresponding uncertain data items.

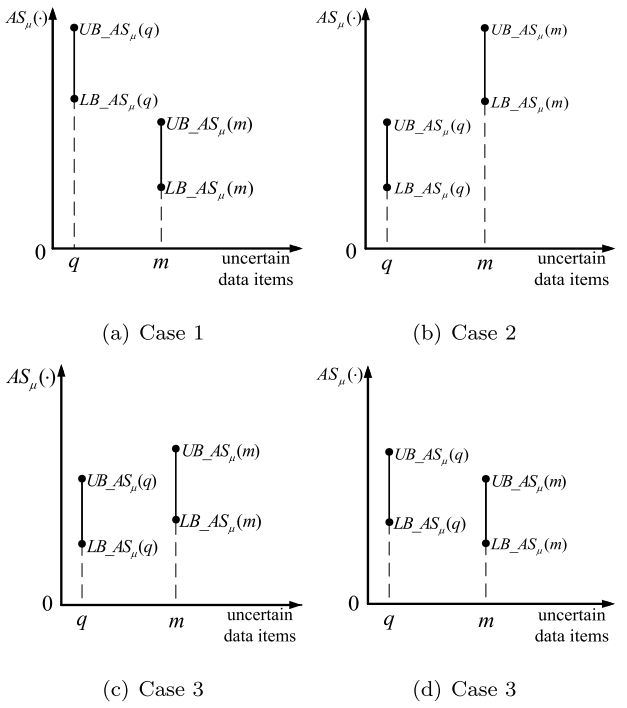


Fig. 4. Aggregated score relationships between the query object q and an MBR m.

- **Case 1:** If $UB\_AS_{m_\mathcal{V}}(m) < LB\_AS_{m_\mathcal{V}}(q)$, as shown in Fig. 4(a), this means that, for $\forall \mu \in m_\mathcal{V}$, $AS_\mu(p)$ of each uncertain object $p \in m$ is smaller than that of the query object q.
- **Case 2:** If $UB\_AS_{m_\mathcal{V}}(q) < LB\_AS_{m_\mathcal{V}}(m)$, as depicted in Fig. 4(b), then for $\forall \mu \in m_\mathcal{V}$, $AS_\mu(q)$ of the query object q is smaller than that of each uncertain object in m.
- **Case 3:** As illustrated in Fig. 4(c), (d), if none of the above two cases holds, then the aggregated scores of q and m are incomparable and there exists an overlap between their score intervals. For $\forall \mu \in m_\mathcal{V}$, it is impossible to determine whether q has a better or worse aggregated score than the object $p \in m$.

In the following, we will propose the pruning heuristics based on the three cases above. First, the concept of dominance is given as follows.

Definition 7 *(Dominance [10]).* Given $\mathcal{D}$, $\forall o, t \in \mathcal{D}$, o dominates t, denoted by $o \prec t$, satisfying $\cap_{j=1}^{d}(o[j] \le t[j])) \cap (\cup_{j=1}^{d}(o[j] < t[j]))$.

Theorem 1. *Given $\mathcal{D}$ and $\mathcal{U}$, $\forall p, q \in \mathcal{D}$, if $p \prec q$ and $Pr(p) \ge Pr(q)$, then for $\forall \mu \in \mathcal{U}$, $Pr_\mu^k(p) \ge Pr_\mu^k(q)$.*

Proof. From Definition 2, we know that

$$Pr_\mu^k(o) = \sum_{\mathcal{W} \in \Omega,\, o \in TOP_k} Pr(\mathcal{W}).$$

For $\forall \mathcal{W} \in \Omega$, if $o \in \mathcal{W}$ and there are at most $k-1$ objects with lower aggregated scores in $\mathcal{W}$, then we can get $o \in TOP_k$. Thus,

$$Pr_\mu^k(o) = Pr(o) \cdot \sum_{\substack{\mathcal{W} \in \Omega \\ |\{t | t \in \mathcal{W}, AS_\mu(t) \le AS_\mu(o)\}| \le k-1}} Pr(\mathcal{W}). \tag{20}$$

Accordingly, we have

$$Pr_\mu^k(p) = Pr(p) \cdot \sum_{\substack{\mathcal{W} \in \Omega \\ |\{t | t \in \mathcal{W}, AS_\mu(t) \le AS_\mu(p)\}| \le k-1}} Pr(\mathcal{W}), \tag{21}$$

$$Pr^k_\mu(q) = Pr(q) \cdot \sum_{\substack{\mathcal{W}\in\Omega \\ |\{t|t\in\mathcal{W}, AS_\mu(t)\le AS_\mu(q)\}|\le k-1}} Pr(\mathcal{W}). \tag{22}$$

As $p \prec q$, we have, for $\forall j\ (1 \le j \le d)$, $p[j] \le q[j]$ and $\exists j\ (1 \le j \le d)$, $p[j] < q[j]$. Then, for $\forall \mu \in \mathcal{U}$, $AS_\mu(p) \le AS_\mu(q)$ obviously. So the event $\{|\{t|t \in \mathcal{W}, AS_\mu(t) \le AS_\mu(p)\}| \le k-1\}$ always contains the event $\{|\{t|t \in \mathcal{W}, AS_\mu(t) \le AS_\mu(q)\}| \le k-1\}$, and corresponding possible worlds have the same inclusion relation. Consequently, we have

$$\sum_{\substack{\mathcal{W}\in\Omega \\ |\{t|t\in\mathcal{W}, AS_\mu(t)\le AS_\mu(p)\}|\le k-1}} Pr(\mathcal{W}) \quad \ge \quad \sum_{\substack{\mathcal{W}\in\Omega \\ |\{t|t\in\mathcal{W}, AS_\mu(t)\le AS_\mu(q)\}|\le k-1}} Pr(\mathcal{W}). \tag{23}$$

As $Pr(p) \ge Pr(q)$, thus for $\forall \mu \in \mathcal{U}$, $Pr^k_\mu(p) \ge Pr^k_\mu(q)$. □

Lemma 4. *Given $\mathcal{D}$, μ, $q \in \mathcal{D}$, and α. For $\forall p \in \mathcal{D}$, if $p \prec q$ (or $AS_\mu(p) \le AS_\mu(q)$), $Pr(p) \ge Pr(q)$, and $Pr^k_\mu(p) \le \alpha$, then $\mu \notin bPRT_k(q)$.*

Proof. Since $p \prec q$ (or $AS_\mu(p) \le AS_\mu(q)$, since $p \prec q \Rightarrow AS_\mu(p) \le AS_\mu(q)$) and $Pr(p) \ge Pr(q)$, according to Theorem 1, for $\forall \mu \in \mathcal{U}$, $Pr^k_\mu(p) \ge Pr^k_\mu(q)$. If $Pr^k_\mu(p) < \alpha$, then $Pr^k_\mu(q) < \alpha$, thereby $q \notin PTT_k$. Therefore, $\mu \notin bPRT_k(q)$. □

Theorem 2. *Given $\mathcal{D}$, $q \in \mathcal{D}$, α, a set of user preferences $\mathcal{V} \subseteq \mathcal{U}$ represented by an MBR $m_\mathcal{V}$, and a set of objects in $\mathcal{D}$ represented by an MBR m. If $UB\_AS_{m_\mathcal{V}}(m) < LB\_AS_{m_\mathcal{V}}(q)$, $Pr(m.u) > Pr(q)$, and $Pr^k_{m_\mathcal{V}.u}(m.u) < \alpha$ or $Pr^k_{m_\mathcal{V}.l}(m.l) < \alpha$, then $m_\mathcal{V}$ can be safely pruned, i.e., no user preference $\mu \in m_\mathcal{V}$ belongs to BPRT query of q answer.*

Proof. Since $UB\_AS_{m_\mathcal{V}}(m) < LB\_AS_{m_\mathcal{V}}(q)$, we have, for $\forall p \in m$, $\mu \in m_\mathcal{V}$, $AS_\mu(p) < AS_\mu(q)$. Especially, for the upper-right corner $m.u$ of m, $AS_\mu(m.u) < AS_\mu(q)$. As $Pr(m.u) > Pr(q)$, for $\forall \mu \in m_\mathcal{V}$, we have $Pr^k_\mu(m.u) \ge Pr^k_\mu(q)$ based on Theorem 1. In particular, for the two opposite corners $m_\mathcal{V}.u$ and $m_\mathcal{V}.l$ of $m_\mathcal{V}$, we have $Pr^k_{m_\mathcal{V}.u}(m.u) \ge Pr^k_\mu(q)$, $Pr^k_{m_\mathcal{V}.l}(m.u) \ge Pr^k_\mu(q)$. Then, if $Pr^k_{m_\mathcal{V}.u}(m.u) < \alpha$ or $Pr^k_{m_\mathcal{V}.l}(m.u) < \alpha$, we can obtain $Pr^k_\mu(q) < \alpha$. Therefore, for $\forall \mu \in m_\mathcal{V}$, $q \notin PTT_k$, thereby $bPRT_k(q) = \emptyset$. Thus $m_\mathcal{V}$ can be safely pruned. □

Theorem 3. *Given $\mathcal{D}$, $q \in \mathcal{D}$, α, a set of user preferences $\mathcal{V} \subseteq \mathcal{U}$ represented by an MBR $m_\mathcal{V}$, and a set of objects in $\mathcal{D}$ represented by an MBR m. If $UB\_AS_{m_\mathcal{V}}(q) < LB\_AS_{m_\mathcal{V}}(m)$, $Pr(q) > Pr(m.l)$, and $Pr^k_{m_\mathcal{V}.l}(m.l) \ge \alpha$ or $Pr^k_{m_\mathcal{V}.u}(m.l) \ge \alpha$, then for $\forall \mu \in m_\mathcal{V}$, $\mu \in bPRT_k(q)$.*

Proof. Since $UB\_AS_{m_\mathcal{V}}(q) < LB\_AS_{m_\mathcal{V}}(m)$, for $\forall p \in m$, $\mu \in m_\mathcal{V}$, $AS_\mu(q) < AS_\mu(p)$. Especially, for the lower-left corner $m.l$ of m, $AS_\mu(q) < AS_\mu(m.l)$. As $Pr(q) > Pr(m.l)$, for $\forall \mu \in m_\mathcal{V}$, we have $Pr^k_\mu(q) \ge Pr^k_\mu(m.l)$ based on Theorem 1. Particularly, for the two opposite corners $m_\mathcal{V}.u$ and $m_\mathcal{V}.l$ of $m_\mathcal{V}$, we have $Pr^k_\mu(q) \ge Pr^k_{m_\mathcal{V}.u}(m.l)$, $Pr^k_\mu(q) \ge Pr^k_{m_\mathcal{V}.l}(m.l)$. Then, if $Pr^k_{m_\mathcal{V}.l}(m.l) \ge \alpha$ or $Pr^k_{m_\mathcal{V}.u}(m.l) \ge \alpha$, we can obtain $Pr^k_\mu(q) \ge \alpha$. Therefore, for $\forall \mu \in m_\mathcal{V}$, $\mu \in bPRT_k(q)$. □

The aforementioned pruning heuristics guide the design of an efficient BPRT query algorithm for determining whether all user preferences in $\mathcal{V} \subseteq \mathcal{U}$ or none of them belong to BPRT query results of q. In the following, we will seamlessly integrate the pruning heuristics into our BPRT query processing algorithm.

5.3. BPRT query processing

In this section, we present the detailed algorithm for BPRT query processing. Specifically, given two data sets $\mathcal{D}$ and $\mathcal{U}$ indexed by a PR-tree $\mathcal{I}_\mathcal{D}$ and an R-tree $\mathcal{I}_\mathcal{U}$, respectively, the BPRT query retrieves a subset of $\mathcal{U}$ so that the query object belongs to the PTT query result set.

Algorithm 2 describes the BPRT query algorithm. Intuitively, as our algorithm traverses indexes $\mathcal{I}_\mathcal{D}$ and $\mathcal{I}_\mathcal{U}$, it reduces the search space of BPRT query by discarding MBRs of user preferences that cannot contribute to the answer set. Essentially, each time an entry of the R-tree is processed, our algorithm tests whether the users that are enclosed by the MBR of the entry, forming a set $\mathcal{V} \subseteq \mathcal{U}$, may belong to the result set or whether it can be pruned. The goal of BPRT query is to expand as few entries as possible by either removing entries of the R-tree or by adding them to the answer set immediately.

In order to accept entries, we maintain two heaps $\mathcal{H}_\mathcal{D}$ and $\mathcal{H}_\mathcal{U}$ for indexes $\mathcal{I}_\mathcal{D}$ and $\mathcal{I}_\mathcal{U}$, respectively, where $\mathcal{H}_\mathcal{D}$ is sorted in descending order of the maximum probability of entry in $\mathcal{I}_\mathcal{D}$, $\mathcal{H}_\mathcal{U}$ is sorted on the Euclidean distance of each entry from objects (line 1). Specifically, we keep three sets *RES*, R, and R_{rfn}, which are initialized to empty (line 2). Set *RES* is used to store the BPRT query results; R stores the objects for computing the top-k probability in the leaf nodes of the PR-tree; R_{rfn} contains MBRs from $\mathcal{I}_\mathcal{D}$ for pruning the user preferences in $\mathcal{I}_\mathcal{U}$.

The algorithm accesses nodes in both $\mathcal{I}_\mathcal{D}$ and $\mathcal{I}_\mathcal{U}$. Firstly, the roots $root(\mathcal{I}_\mathcal{D})$ and $root(\mathcal{I}_\mathcal{U})$ of $\mathcal{I}_\mathcal{D}$ and $\mathcal{I}_\mathcal{U}$ are inserted into heaps $\mathcal{H}_\mathcal{D}$ and $\mathcal{H}_\mathcal{U}$, respectively (line 3); Then, each time we pop out a heap entry from one of the heaps (line 5 or 7). If the current popped entry $e_\mathcal{D}$ is from heap $\mathcal{H}_\mathcal{D}$, we invoke algorithm ***Node_Handler_D*** (lines 5–6), which obtains

Algorithm 2 BPRT_query processing.

Input:
$\mathcal{I}_{\mathcal{D}}$, $\mathcal{I}_{\mathcal{U}}$, q in $\mathcal{D}$, α, and k
Output:
RES
1: initialize two heaps $\mathcal{H}_{\mathcal{D}}$ and $\mathcal{H}_{\mathcal{U}}$ for accepting entries;
2: $RES \leftarrow \emptyset$, $R \leftarrow \emptyset$, $R_{rfn} \leftarrow \emptyset$;
3: insert entries in roots $root(\mathcal{I}_{\mathcal{D}})$ and $root(\mathcal{I}_{\mathcal{U}})$ into heaps $\mathcal{H}_{\mathcal{D}}$ and $\mathcal{H}_{\mathcal{U}}$, respectively;
4: **while** both $\mathcal{H}_{\mathcal{D}}$ and $\mathcal{H}_{\mathcal{U}}$ are not empty **do**
5: $e_{\mathcal{D}} \leftarrow \mathcal{H}_{\mathcal{D}}.pop()$;
6: $Node\_Handler\_\mathcal{D}(q, e_{\mathcal{D}}, \mathcal{H}_{\mathcal{D}}, R, R_{rfn}, k, \alpha)$;
7: $e_{\mathcal{U}} \leftarrow \mathcal{H}_{\mathcal{U}}.pop()$;
8: $Node\_Handler\_\mathcal{U}(q, e_{\mathcal{U}}, \mathcal{H}_{\mathcal{U}}, R, R_{rfn}, k, \alpha)$;
9: **return** *RES*.

Algorithm 3 Node_Handler_D.

Input:
q, $e_{\mathcal{D}}$, R, $\mathcal{H}_{\mathcal{D}}$, α, and k
Output:
R_{rfn}
1: **if** $Pr(e_{\mathcal{D}}.l) < \alpha$ **then**
2: return;
3: **if** $e_{\mathcal{D}}$ is a leaf node **then**
4: **for** each uncertain object $o \in e_{\mathcal{D}}$ **do**
5: $R \leftarrow R \cup \{o\}$;
6: **else**
7: **for** each entry e_i in $e_{\mathcal{D}}$ **do**
8: $\mathcal{H}_{\mathcal{D}}.push(e_i)$;
9: $R_{rfn} \leftarrow R_{rfn} \cup \{e_i\}$;
10: **return** R_{rfn}.

the uncertain objects and MBRs; otherwise, we invoke algorithm ***Node_Handler_U*** (lines 7–8), which pops out top entry $e_{\mathcal{U}}$ from $\mathcal{H}_{\mathcal{U}}$ and applies the pruning heuristics proposed aforementioned. The iteration stops when one of heaps is empty (line 4). Finally, the algorithm returns result set *RES* (line 9).

Algorithm 3 shows the details of *Node_Handler_D*. Specifically, if the maximum probability of entry $e_{\mathcal{D}}$ is less than the user-specified probability threshold α, then the algorithm terminates (lines 1–2). The reason is, if $Pr(e_{\mathcal{D}}.l) < \alpha$, then $Pr(q) < \alpha$, since the maximum probability of an entry e equals to the maximum existence probability of the elements rooted at e. Based on Lemma 1, $q \notin PTT_k$. So, arbitrary user preference cannot be the BPRT query result. If $e_{\mathcal{D}}$ is a leaf node, we add the objects under $e_{\mathcal{D}}$ to set R for computing the top-k probability of objects (lines 3–5). Otherwise, if $e_{\mathcal{D}}$ is an intermediate node, we scan each entry e_i in $e_{\mathcal{D}}$ and insert them into heap $\mathcal{H}_{\mathcal{D}}$ (lines 7–8). In addition, we also add them to set R_{rfn} for pruning the useless users in Algorithm ***Node_Handler_U*** (line 9).

Then, we invoke **Algorithm 4** to process the nodes in $\mathcal{I}_{\mathcal{U}}$. Specifically, if the top entry $e_{\mathcal{U}}$ from heap $\mathcal{H}_{\mathcal{U}}$ is a leaf node, then we check each user μ_i in $e_{\mathcal{U}}$, and estimate whether μ_i belongs to the BPRT query results based on Lemma 4 (lines 3–6). Otherwise, for the remaining users μ_j in $e_{\mathcal{U}}$, if $Pr^k_{\mu_j}(q) \geq \alpha$, we add μ_j to result set *RES* (lines 7–8). If $e_{\mathcal{U}}$ is an intermediate node (line 9), we scan each entry e_i and m_i in $e_{\mathcal{U}}$ and set R_{rfn}, respectively, then we apply the pruning heuristics based on Theorems 2 and 3 (lines 10–18). Finally, for the remaining entries in $e_{\mathcal{U}}$ which are incomparable, we insert them into the heap $\mathcal{H}_{\mathcal{U}}$ (line 20).

Note that we maintain a dynamic programming table for computing top-k probabilities of objects with cost $O(k|\mathcal{D}|)$ in **Algorithm 5**.

Complexity Analysis. Let $|\mathcal{I}_{\mathcal{D}}|$ and $|\mathcal{I}_{\mathcal{U}}|$ are the levels of indexes $\mathcal{I}_{\mathcal{D}}$ and $\mathcal{I}_{\mathcal{U}}$. Algorithm 3 traverses tree $\mathcal{I}_{\mathcal{D}}$ once with cost $O(\log|\mathcal{I}_{\mathcal{D}}|)$. On the other hand, Algorithm 4 traverses tree $\mathcal{I}_{\mathcal{U}}$ with cost $O(\log|\mathcal{I}_{\mathcal{U}}|)$. Additionally, it takes cost $O\big[k|R|(2|R| + (2d+1)|R_{rfn}|)\big]$ to implement pruning heuristics, where the evaluation of top-k probability of each uncertain object needs cost $O(k|R|)$. Thus, the time complexity of the BPRT query algorithm, in the worst case, is $O\big[\log|\mathcal{I}_{\mathcal{D}}| + k|R|\log|\mathcal{I}_{\mathcal{U}}|(2|R| + (2d+1)|R_{rfn}|)\big]$.

Theorem 4 *(Correctness of BPRT query algorithm). The BPRT query processing algorithm always reports the correct answer set.*

Proof. In our BPRT query algorithm, every entry e of the R-tree indexing users $\mathcal{U}$ is either added to the result set or pruned. Note that expanding an entry also eventually results in adding or removing the enclosed entries. Therefore, it suffices to show that the BPRT query never discards *false negatives*, i.e., an entry that contains a user that belongs to a probabilistic reverse top-k result. On the other hand, it never adds *false positives*, i.e., an entry which contains a user that is not a member of the probabilistic reverse top-k results.

Algorithm 4 Node_Handler_U.

Input:
q, $e_{\mathcal{U}}$, R, R_{rfn}, $\mathcal{H}_{\mathcal{U}}$, α, and k
Output:
RES
1: **if** $e_{\mathcal{U}}$ is a leaf node **then**
2: **for** each preference weighting vector μ_i in $e_{\mathcal{U}}$ **do**
3: **for** each uncertain object $p \in R$ **do**
4: **if** $AS_{\mu_i}(p) < AS_{\mu_i}(q)$ and $Pr(p) > Pr(q)$ **then**
5: **if** $Pr^k_{\mu_i}(p) < \alpha$ **then**
6: prune safely μ_i from $e_{\mathcal{U}}$;
7: **if** $Pr^k_{\mu_i}(q) \geq \alpha$ **then**
8: $RES \leftarrow RES \cup \{\mu_i\}$;
9: **else**
10: **for** each entry e_i in $e_{\mathcal{U}}$ **do**
11: **for** each entry m_i in R_{rfn} **do**
12: **if** $((UB\_AS_{e_i}(m_i) < LB\_AS_{e_i}(q))$ and $(Pr(m_i.u) > Pr(q)))$ **then**
13: **if** $((Pr^k_{e_i.u}(m_i.u) < \alpha)$ or $(Pr^k_{e_i.l}(m_i.u) < \alpha))$ **then**
14: prune safely all user preferences under e_i;
15: **else**
16: **if** $((UB\_AS_{e_i}(q) < LB\_AS_{e_i}(m_i))$ and $(Pr(q) > Pr(m_i.l)))$ **then**
17: **if** $((Pr^k_{e_i.l}(m_i.l) \geq \alpha)$ or $(Pr^k_{e_i.u}(m_i.l) \geq \alpha))$ **then**
18: $RES \leftarrow RES \cup \{\mu_m\}$, $\forall \mu_m \in e_i$;
19: **else**
20: $\mathcal{H}_{\mathcal{U}}.push(e_i)$.

Algorithm 5 Top-k_Probability_Computing.

Input:
$\mathcal{D} = \{o_1, o_2, \ldots, o_{|\mathcal{D}|}\}$ and k
Output:
$T(0 \ldots k; 1 \ldots |\mathcal{D}|)$
1: **for** $i = 1$ to $|\mathcal{D}|$ **do**
2: $T(0, i) = 0$;
3: **for** $k' = 1$ to k **do**
4: $T(k', 1) = Pr(o_1)$;
5: **for** $i = 2$ to $|\mathcal{D}|$ **do**
6: **for** $k' = 1$ to k **do**
7: $T(k', i) = (T(k', i-1) \cdot \frac{1 - Pr(o_{i-1})}{Pr(o_{i-1})} + T(k'-1, i-1)) \cdot Pr(o_i)$;
8: **return** $T(0 \ldots k; 1 \ldots |\mathcal{D}|)$.

- *false negatives:* Algorithm 4 determines to remove an entry e in two cases. In the first case, as shown in Lemma 4, if there is $p \in \mathcal{D}$ such that $AS_\mu(p) \leq AS_\mu(q)$ or $p \prec q$, and $Pr^k_\mu(p) \leq \alpha$, then for $\forall \mu \in \mathcal{U}$, $\mu \notin bPRT_k(q)$. In the second case, based on Theorem 2, if there exists an object set represented by an MBR m such that $UB\_AS_e(m) < LB\_AS_e(q)$, $Pr(m.u) > Pr(q)$, and $Pr^k_{e.u}(m.u) < \alpha$ or $Pr^k_{e.l}(m.u) < \alpha$, then e can be safely pruned.
- *false positives:* According to Theorem 3, Algorithm 4 adds an entry e to the result set, if there is an object set represented by an MBR m such that $UB\_AS_e(q) < LB\_AS_e(m)$, $Pr(q) > Pr(m.l)$, and $Pr^k_{e.l}(m.l) \geq \alpha$ or $Pr^k_{e.u}(m.l) \geq \alpha$, then for $\forall \mu \in e$, $\mu \in bPRT_k(q)$. □

6. Experimental evaluation

In this section, we demonstrate the efficiency and effectiveness of BPRT queries through a series of experiments over both real-world and synthetic data sets. All experiments are implemented in Microsoft Visual C++ V6.0, and the experiments run on a PC with a 2.93 GHz Intel® Core2 Duo CPU with 4 GB of main memory, and a 500 GB hard disk, running Windows Win7 32 bit Operating System. The block size is set to 4 KB.

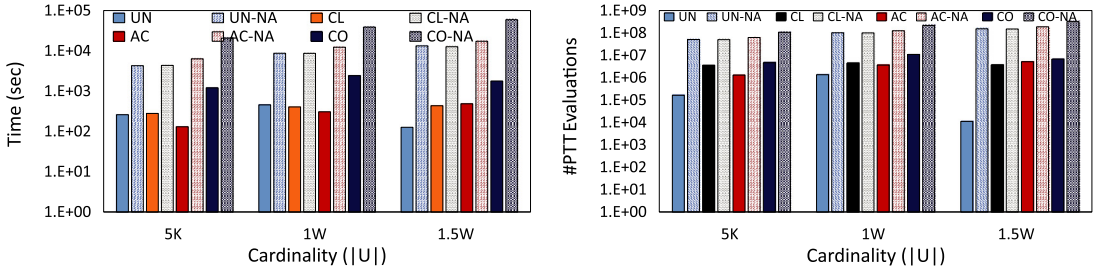
6.1. Experimental setup

Data sets. In the experimental evaluations, as far as the uncertain data set $\mathcal{D}$ is concerned, both real-world and synthetic data sets are used. In the case of synthetic data sets, uniform (UN), correlated (CO), anti-correlated (AC), and clustered (CL) data are generated, respectively, and the generated values of each attribute belong to $[0, 1]$. For the UN data set, all attribute values are generated independently based on a uniform distribution. The CO, AC, and CL data sets are generated as illustrated in [39,43]. For the data set $\mathcal{U}$, UN and CL are examined.

Two real-world data sets, namely *NBA* and *Cars*, are also applied to evaluate the efficiency and effectiveness of BPRT queries in real-world applications. Specially, the *NBA* contains 17,265 5-dimensional points, representing the average statistics of a player per year in the number of points scored, rebounds, assists, steals, and blocks. Like [10], we use uniform distribution to randomly generate a presence probability for each point to make them be uncertain. In addition, *Cars* data

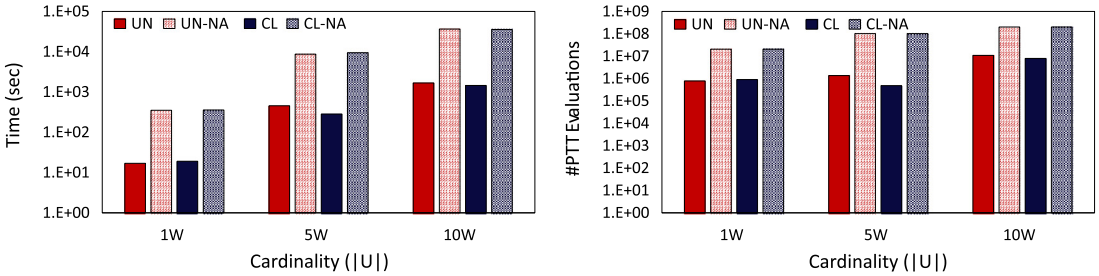
Table 7
Experimental parameters.

Parameters	Values	Parameters	Values
$\mathcal{D}$	**UN**, CO, AC, CL	k	10, **50**, 100
$\mathcal{U}$	**UN**, CL	α	0.2, **0.4**, 0.6, 0.8
d	2, **4**, 6, 8	$C_{\mathcal{D}}$, $C_{\mathcal{U}}$	5
$\lvert\mathcal{D}\rvert$	1W, **5W**, 10W	$\sigma^2_{\mathcal{D}}$, $\sigma^2_{\mathcal{U}}$	0.05^2
$\lvert\mathcal{U}\rvert$	5K, **1W**, 1.5W	#queries	10



(a) Time. (b) #PTT evaluations.

Fig. 5. Comparative performance with NA for varying $|\mathcal{U}|$.



(a) Time. (b) #PTT evaluations.

Fig. 6. Comparative performance with NA for varying $|\mathcal{D}|$.

set is used in [33,15] which consists of 4-dimensional data objects with a cardinality 45,311 capturing several properties of each car such as id, mileage, make year, and price. We only consider two numerical attributes of mileage and price. In [33], to convert the data set into an uncertain data set, the authors obtain each vehicle's available probability with a hidden Markov model (HMM). The transition graph of HMM assumes an independent selling probability for each vehicle, and assumes that once the car is sold, it shall not return to the available state.

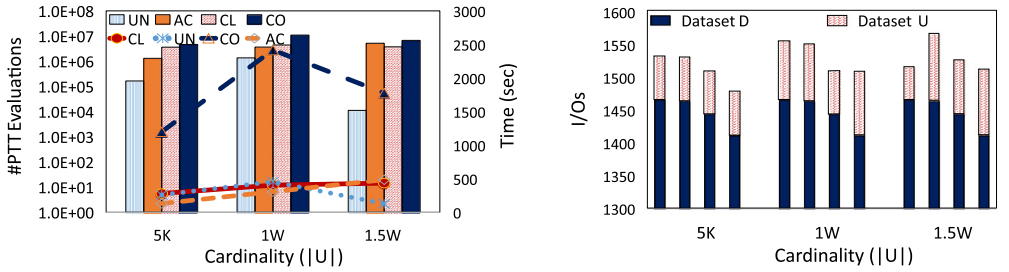
Metrics. Our primary performance metrics include: 1) the query execution time required by the query, 2) the I/Os used, and 3) the total number of PTT evaluations conducted. Particularly, in the figures showing I/Os, each bar corresponds to the total number of I/Os, while the dark red part of the bar shows the I/Os induced on $\mathcal{U}$ and the dark blue part of the bar shows the I/O costs induced on $\mathcal{D}$.

Parameters. The experimental parameters as well as their values to be used in our experiments are summarized in Table 7. Unless specifically pointed out, each parameter is set to its default value as illustrated in bold in the table. For the clustered data set, the cluster centroids $C_{\mathcal{D}}$ and $C_{\mathcal{U}}$, as well as the variances $\sigma_{\mathcal{D}}$ and $\sigma_{\mathcal{U}}$ are also set to 5 and 0.05^2 by default, respectively. Each reported result is an average of 10 queries under the same parameter setting.

6.2. Performance comparisons with NA

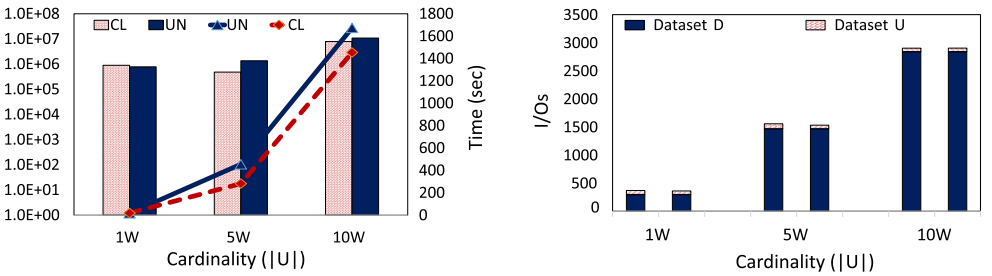
As mentioned before, the BPRT query, if computed in a straightforward manner without any pruning heuristic (***naïve algorithm, NA, for short***), requires evaluating a PTT query for each μ in $\mathcal{U}$, which is prohibitively expensive and does not scale even for moderate data sets, let alone the big $\mathcal{U}$ and $\mathcal{D}$.

The comparative performance between the BPRT query algorithm and *NA* is illustrated in Figs. 5 and 6. Note that the y-axis where the numbers are displayed in E-notation is in logarithmic scale, similarly hereinafter. From these pictures, we can know that the query execution time and the number of PTT queries of *NA* increase (nearly two orders of magnitude) distinctly compared with the BPRT query algorithm under various experimental settings regardless of data distribution, respectively, which shows the proposed BPRT query algorithm based on pruning heuristics has unexceptionable preference compared with *NA*. For example, for UN data set, the running time is 8,688.846*s*, the number of PTT evaluations is



(a) #PTT evaluations & Time. (b) I/Os.

Fig. 7. Performance for varying $|\mathcal{U}|$ under different $\mathcal{D}$.



(a) #PTT evaluations & Time. (b) I/Os.

Fig. 8. Performance for varying $|\mathcal{D}|$ under different $\mathcal{U}$.

101,880,000, and I/Os for $\mathcal{U}$ of *NA* is 135 for $|\mathcal{D}| = 5$W, $|\mathcal{U}| = 1$W, $k = 50$, and $\alpha = 0.4$, while for the BPRT query algorithm, they are 454.7284*s*, 1,360,958, and 90, respectively. Due to large costs of *NA*, we only evaluate preference of BPRT queries under various settings in the following sections. The pruning effects of proposed pruning heuristics for different experimental parameters are shown in Tables 8–11 in Section 6.7.

6.3. Performance with data set size

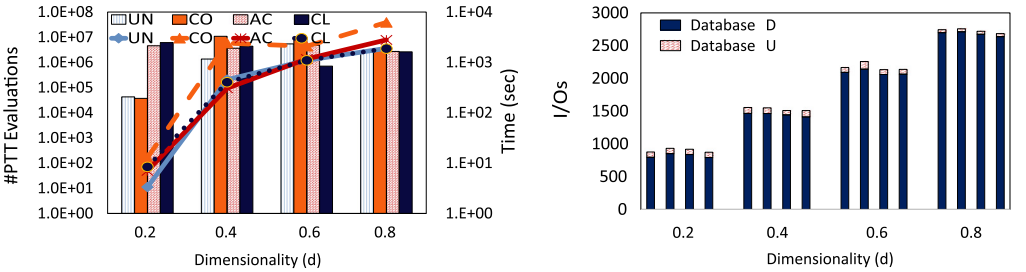
In this series of experiments, the comparative performance of the BPRT query algorithm for different data distributions is evaluated by varying the cardinality of data sets.

Figs. 7 and 8 present experimental results for varying $\mathcal{D}$ and $\mathcal{U}$. Fig. 7(a) and Fig. 8(a) depicts the query execution time (the secondary y-axis) and the number of PTT evaluations (the principal y-axis) of the algorithm. From these pictures, we can see that, the query execution time increases generally with the cardinality of data set, which is intuitive. Depending on the data distribution, the query time is different for various data sets, which shows the scalability of the BPRT query algorithm. Particularly, it takes more time for the algorithm to handle with CO data set. This can be explained based on the fact that for the data of correlation, objects of large attribute values are attributed with high probability, which allows pruning a small number of users represented by an MBR $m_{\mathcal{V}}$ based on Theorem 2, while for the other data distributions such as AC data, more MBRs $m_{\mathcal{V}}$ can be pruned before concluding the BPRT query results.

We also notice that the CO data requires more PTT evaluations than the other data distributions. One reason is that the cardinality of BPRT query result set is high so that a large number of objects are in the PTT query answer set for many users. Another reason is that the algorithm needs to evaluate more PTT queries during the pruning phase. The I/O costs conducted by the BPRT query algorithm are described in Fig. 7(b) and Fig. 8(b). The four bars in Fig. 7(b) represent UN, CO, AC, and CL data, respectively; and the two bars in Fig. 8(b) denote UN and CL data, respectively, similarly hereinafter. Intuitively, the I/Os increase with increasing $|\mathcal{U}|$ and $|\mathcal{D}|$, and the I/Os are different for different data distributions. Obviously, the proposed algorithm based on the R-tree and PR-tree index has optimal I/O costs compared with sequential scan without an index. Notice that the answer set of the BPRT query for an uncertain query point may be empty. However, it is also very informative for a product manufacturer, because this indicates that the particular product is not popular for any user, compared to their competitors' products.

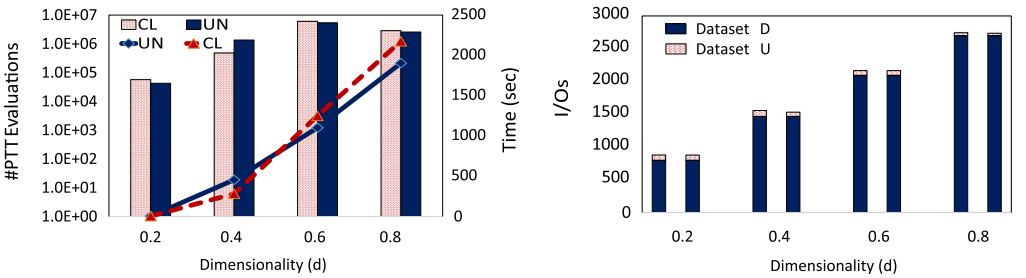
6.4. Performance with dimensionality

The comparative performance of the proposed algorithm for increasing dimensionality under different $\mathcal{D}$ and $\mathcal{U}$ is depicted in Figs. 9 and 10. We can know from the figures that the query execution time increases with d, which is intuitive. It is different for different data distributions, which shows the scalability of the BPRT query algorithm *w.r.t.* $\mathcal{D}$ and $\mathcal{U}$. Intuitively, the I/Os increase as d increases. This shows that the performance of R-tree and PR-tree could decrease with an



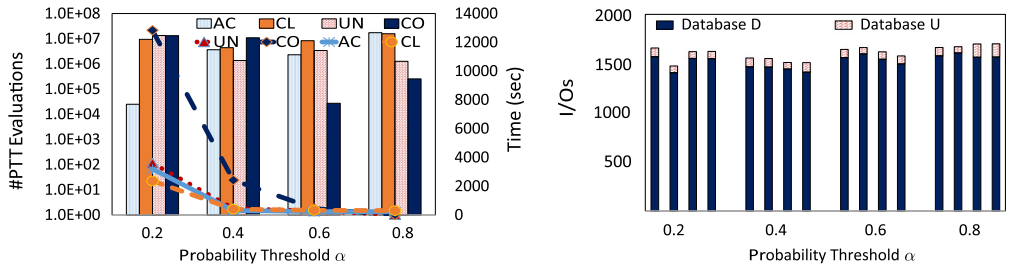
(a) #PTT evaluations & Time. (b) I/Os.

Fig. 9. Performance for varying d under different $\mathcal{D}$.



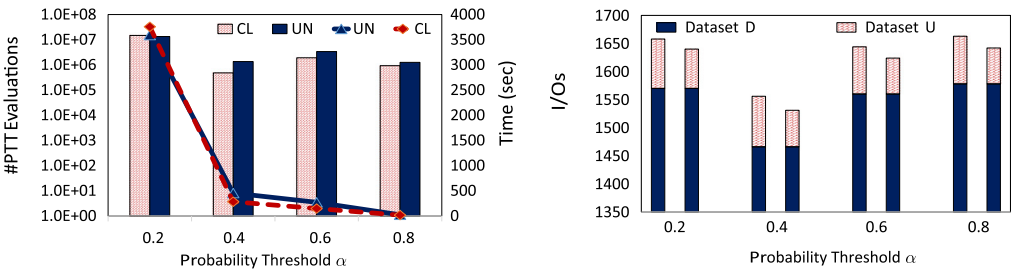
(a) #PTT evaluations & Time. (b) I/Os.

Fig. 10. Performance for varying d under different $\mathcal{U}$.



(a) #PTT evaluations & Time. (b) I/Os.

Fig. 11. Performance for varying α under different $\mathcal{D}$.



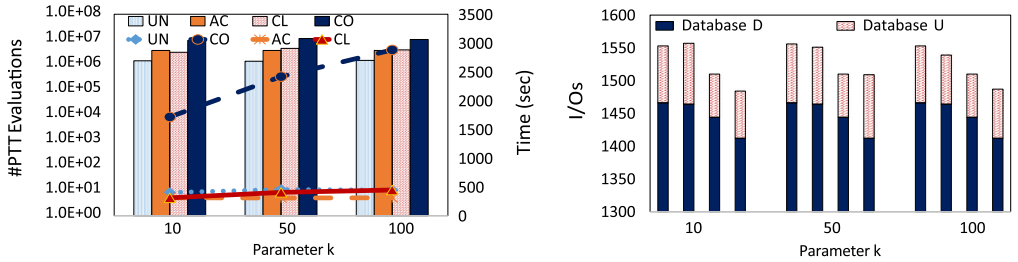
(a) #PTT evaluations & Time. (b) I/Os.

Fig. 12. Performance for varying α under different $\mathcal{U}$.

increasing dimensionality. In terms of the number of total top-k evaluations, we observe that it takes more PTT evaluations for CO data because of the reasons aforementioned. In addition, it needs to evaluate more PTT queries for dimensionality $d = 8$, which shows the poor performance for R-tree and PR-tree in high-dimensional case.

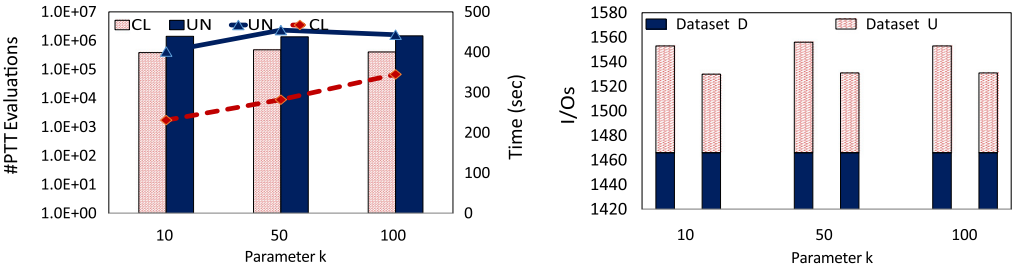
6.5. Performance with probability threshold

Figs. 11 and 12 show the comparative performance of the proposed algorithm for increasing α under various $\mathcal{D}$ and $\mathcal{U}$. We can see from the figures that the query execution time, the number of PTT queries, and I/Os of the algorithm decrease



(a) #PTT evaluations & Time. (b) I/Os.

Fig. 13. Performance for varying k under different $\mathcal{D}$.



(a) #PTT evaluations & Time. (b) I/Os.

Fig. 14. Performance for varying k under different $\mathcal{U}$.

as α increases, respectively. This is because, as α increases, it allows pruning a large number of users represented by an MBR $m_{\mathcal{V}}$. Depending on the data distribution, the execution time is different for different $\mathcal{D}$ and $\mathcal{U}$, which shows the scalability of the algorithm with respect to $\mathcal{D}$ and $\mathcal{U}$. We observe that the comparative performance on CO data is worse than the other data distributions for the reasons aforementioned. As such, in terms of the I/Os illustrated in Fig. 11(b) and Fig. 12(b), the proposed algorithm based indexes has optimal I/Os compared with sequential access without any index.

6.6. Performance with parameter k

In this series of experiments, we evaluate the comparative performance of the proposed algorithm for increasing k value under different $\mathcal{D}$ and $\mathcal{U}$. Fig. 13 depicts the experiment results for different $\mathcal{D}$. Fig. 14 illustrates the performance of the algorithm for increasing k under various $\mathcal{U}$. As illustrated in these charts, the BPRT query algorithm has good scalability as k increases.

6.7. Pruning effects

Tables 8–11 depict the pruning effects (represented by *pruning ratio* (PR)) of the proposed pruning heuristics under various $\mathcal{D}$ and $\mathcal{U}$, where

$$PR = \frac{|\mathcal{U}| - |\mathcal{U}_{aft}|}{|\mathcal{U}|} \times 100\%, \tag{24}$$

$|\mathcal{U}_{aft}|$ is the number of $\mathcal{U}$ after pruning based on the pruning heuristics. We can know from the tables that the pruning heuristics have an unexceptionable pruning efficiency with the pruning ratio around 95%, which can prune the vast majority of user preferences, thus reducing largely the search space of the BPRT query and being efficient for uncertain big data processing.

6.8. Performance with real data set

In this series of experiments, we assess BPRT query algorithm for increasing α and k on real-life data set. The results are illustrated in Figs. 15 and 16. The pruning ratios of pruning heuristics for the two data sets are no less than 97%. We can know from the pictures that the comparative performance of the proposed algorithm is in accordance with the case of synthetic data under the same parameter settings, which shows the scalability and effectiveness of our proposed approaches.

Table 8
Pruning ratio for varying $|\mathcal{U}|$ and d under various $\mathcal{D}$ (%).

$\mathcal{D}$	Arg.						
	$\mathcal{U}$			d			
	5K	1W	1.5W	2	4	6	8
UN	93.88	94.85	99.15	95.27	94.85	98.17	99.12
CO	94.16	94.90	96.84	94.76	94.90	96.58	99.12
AC	97.88	97.07	97.24	96.69	97.07	98.46	99.12
CL	92.76	94.53	96.88	93.93	94.53	98.17	99.12

Table 9
Pruning ratio for varying α and k under various $\mathcal{D}$ (%).

$\mathcal{D}$	Arg.						
	α				k		
	0.2	0.4	0.6	0.8	10	50	100
UN	95.11	94.85	94.34	92.96	95.03	94.85	95.42
CO	95.90	94.90	95.94	96.02	95.35	94.90	95.33
AC	96.69	97.07	95.79	96.01	97.07	97.07	97.07
CL	96.85	94.53	83.57	92.58	95.76	94.53	95.41

Table 10
Pruning ratio for varying $|\mathcal{D}|$ and d under various $\mathcal{U}$ (%).

$\mathcal{D}$	Arg.						
	$\mathcal{D}$			d			
	1W	5W	10W	2	4	6	8
UN	95.17	94.85	94.62	95.27	94.85	98.17	99.12
CL	94.47	96.33	95.23	93.68	96.33	97.95	99.02

Table 11
Pruning ratio for varying α and k under various $\mathcal{U}$ (%).

$\mathcal{D}$	Arg.						
	α				k		
	0.2	0.4	0.6	0.8	10	50	100
UN	95.11	94.85	94.34	92.96	95.03	94.85	95.42
CL	95.06	96.33	97.14	95.26	96.99	96.33	96.85

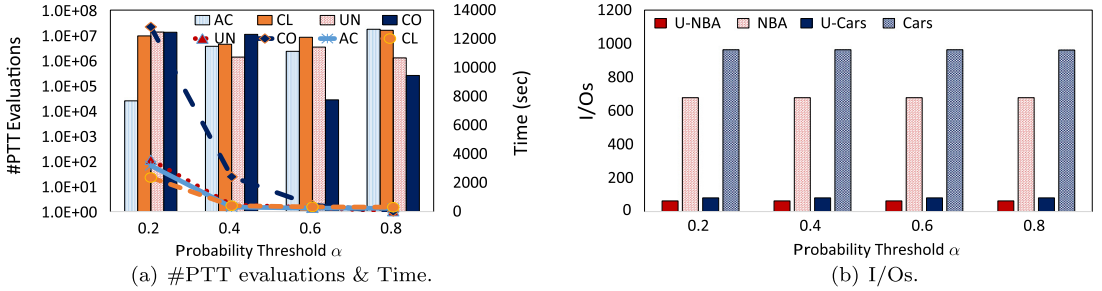
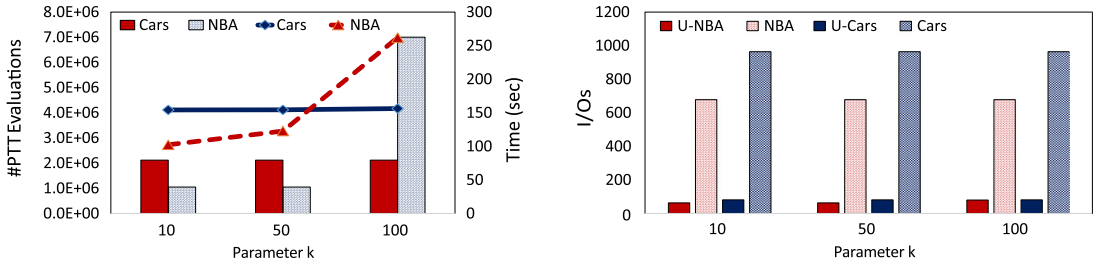


Fig. 15. Comparative performance of *NBA/Cars* data set for varying probability threshold α.

7. Conclusions

There has been an increasing growth in numerous applications that naturally generate large volumes of uncertain data. By the advent of such applications, the support of advanced analysis query processing such as the top-k and reverse top-k for uncertain big data has become important. However, existing approaches for reverse top-k queries are all based on the assumption that the underlying data are exact (or certain). Due to the intrinsic differences between uncertain and certain data, these methods cannot be applied to uncertain cases directly. Motivated by this, in this paper, we firstly present two versions of probabilistic reverse top-k queries, namely *monochromatic* and *bichromatic*, in the context of uncertain data. Then we analyze the solution space of MPRT queries. Thereafter, we present an efficient algorithm for answering BPRT queries.



(a) #PTT evaluations & Time. (b) I/Os.

Fig. 16. Comparative performance of *NBA/Cars* data set for varying parameter k.

Furthermore, in order to reduce the search space of query processing, several effective pruning heuristics with the pruning ratio around 95% are presented, which could discard candidate users without evaluating the associated PTT queries and be efficient for uncertain big data query processing. Extensive experiments have demonstrated the efficiency and effectiveness of our proposed approaches.

As for our future work, it is important and interesting to explore in detail the solving for linear matrix inequalities for answering MPRT queries. In addition, we are going to study approximate probabilistic reverse top-k query algorithms that return quickly a good approximation of results. Particularly, in order to further improve query efficiency, we plan to extend our approaches to parallel distributed query processing for the future work.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their valuable and helpful comments on improving the manuscript. The research was partially funded by the Key Program of National Natural Science Foundation of China (Grant Nos. 61133005, 61432005), the National Natural Science Foundation of China (Grant Nos. 61370095, 61472124), the International Science & Technology Cooperation Program of China (Grant No. 2015DFA11240), the National High-tech R&D Program of China (Grant No. 2015AA015303), the Hunan Provincial Innovation Foundation for Postgraduate (Grant No. CX2016B066), and the Outstanding Graduate Student Innovation Fund Program of Collaborative Innovation Center of High Performance Computing.

References

[1] C.C. Aggarwal, P.S. Yu, A survey of uncertain data algorithms and applications, IEEE Trans. Knowl. Data Eng. 21 (5) (2009) 609–623.
[2] D. Barbará, H. Garcia-Molina, D. Porter, The management of probabilistic data, IEEE Trans. Knowl. Data Eng. 4 (5) (1992) 487–502.
[3] M.A. Cheema, X.M. Lin, W. Wang, W.J. Zhang, J. Pei, Probabilistic reverse nearest neighbor queries on uncertain data, IEEE Trans. Knowl. Data Eng. 22 (4) (2010) 550–564.
[4] J.C. Chen, R. Cheng, Efficient evaluation of imprecise location-dependent queries, in: Proc. of the 23rd Int. Conf. on Data Engineering, ICDE, IEEE, 2007, pp. 586–595.
[5] L. Chen, M.T. Özsu, V. Oria, Robust and fast similarity search for moving object trajectories, in: Proc. of the ACM Int. Conf. on Management of Data, SIGMOD, ACM, 2005, pp. 491–502.
[6] R. Cheng, Y. Zhang, E. Bertino, S. Prabhakar, Preserving user location privacy in mobile data management infrastructures, in: Privacy Enhancing Technologies, Springer, 2006, pp. 393–412.
[7] G. Cormode, F. Li, K. Yi, Semantics of ranking queries for probabilistic data and expected ranks, in: Proc. of the 25th Int. Conf. on Data Engineering, ICDE, IEEE, 2009, pp. 305–316.
[8] N. Dalvi, C. Re, D. Suciu, Queries and materialized views on probabilistic databases, J. Comput. Syst. Sci. 77 (3) (2011) 473–490.
[9] A. Das Sarma, O. Benjelloun, A. Halevy, J. Widom, Working models for uncertain data, in: Proc. of the 22nd Int. Conf. on Data Engineering, ICDE, IEEE, 2006, p. 7.
[10] X.F. Ding, H. Jin, Efficient and progressive algorithms for distributed skyline queries over uncertain data, IEEE Trans. Knowl. Data Eng. 24 (8) (2012) 1448–1462.
[11] R. Fagin, R. Kumar, D. Sivakumar, Comparing top k lists, SIAM J. Discrete Math. 17 (1) (2003) 134–160.
[12] A. Faradjian, J. Gehrke, P. Bonnet, GADT: a probability space ADT for representing and querying the physical world, in: Proc. of the 18th Int. Conf. on Data Engineering, ICDE, IEEE, 2002, pp. 201–211.
[13] S. Flesca, F. Furfaro, F. Parisi, Consistency checking and querying in probabilistic databases under integrity constraints, J. Comput. Syst. Sci. 80 (7) (2014) 1448–1489.
[14] Y.J. Gao, Q. Liu, B.H. Zheng, G. Chen, On efficient reverse skyline query processing, Expert Syst. Appl. 41 (7) (2014) 3237–3249.
[15] Y.J. Gao, Q. Liu, B.H. Zheng, L. Mou, G. Chen, Q. Li, On processing reverse k-skyband and ranked reverse skyline queries, Inf. Sci. 293 (2015) 11–34.
[16] S. Ge, U. Hou, N. Mamoulis, D.W. Cheung, et al., Efficient all top-k computation: a unified solution for all top-k, reverse top-k and top-m influential queries, IEEE Trans. Knowl. Data Eng. 25 (5) (2013) 1015–1027.
[17] T.J. Ge, S. Zdonik, S. Madden, Top-k queries on uncertain data: on score distribution and typical answers, in: Proc. of the ACM Int Conf. on Management of Data, SIGMOD, ACM, 2009, pp. 375–388.
[18] B. Gedik, K.L. Wu, P.S. Yu, L. Liu, Processing moving queries over moving objects using motion-adaptive indexes, IEEE Trans. Knowl. Data Eng. 18 (5) (2006) 651–668.
[19] Q.D. Guo, M.S.E. Din, L.H. Zhi, Computing rational solutions of linear matrix inequalities, in: Proc. of the 38th Int. Symp. on Symbolic and Algebraic Computation, ISSAC, 2013, pp. 197–204.
[20] M. Hua, J. Pei, X.M. Lin, Ranking queries on uncertain data, VLDB J. 20 (1) (2011) 129–153.

[21] M. Hua, J. Pei, W.J. Zhang, X.M. Lin, Efficiently answering probabilistic threshold top-k queries on uncertain data, in: Proc. of the 24th Int. Conf. on Data Engineering, ICDE, IEEE Computer Society Press, 2008, pp. 1403–1405.
[22] M. Hua, J. Pei, W.J. Zhang, X.M. Lin, Ranking queries on uncertain data: a probabilistic threshold approach, in: Proc. of the ACM Int. Conf. on Management of Data, SIGMOD, ACM, 2008, pp. 673–686.
[23] J. Jestes, G. Cormode, F.F. Li, K. Yi, Semantics of ranking queries for probabilistic data, IEEE Trans. Knowl. Data Eng. 23 (12) (2011) 1903–1917.
[24] C.Q. Jin, R. Zhang, Q.Q. Kang, Z. Zhang, A.Y. Zhou, Probabilistic reverse top-k queries, in: Proc. of the 19th Int. Conf. on Database Systems for Advanced Applications, DASFAA, Springer, 2014, pp. 406–419.
[25] J.L. Koh, L.Y. Chen, A.L.P. Chen, Finding k most favorite products based on reverse top-t queries, VLDB J. 23 (4) (2014) 541–564.
[26] J. Li, B. Saha, A. Deshpande, A unified approach to ranking in probabilistic databases, Proc. VLDB Endow. 2 (1) (2009) 502–513.
[27] M. Li, Y. Liu, Underground coal mine monitoring with wireless sensor networks, ACM Trans. Sens. Netw. 5 (2) (2009) 10.
[28] X. Lian, L. Chen, Probabilistic top-k dominating queries in uncertain databases, Inf. Sci. 226.
[29] X. Lian, L. Chen, Monochromatic and bichromatic reverse skyline search over uncertain databases, in: Proc, of the ACM Int. Conf. on Management of Data, SIGMOD, ACM, 2008, pp. 213–226.
[30] X. Lian, L. Chen, Probabilistic ranked queries in uncertain databases, in: Proc. of the 11th Int. Conf. on Extending Database Technology: Advances in Database Technology, EDBT, ACM, 2008, pp. 511–522.
[31] X. Lian, L. Chen, Top-k dominating queries in uncertain databases, in: Proc. of the 12th Int. Conf. on Extending Database Technology: Advances in Database Technology, EDBT, ACM, 2009, pp. 660–671.
[32] X. Lian, L. Chen, Shooting top-k stars in uncertain databases, VLDB J. 20 (6) (2011) 819–840.
[33] X.J. Liu, D.N. Yang, M. Ye, W.C. Lee, U-skyline: a new skyline query for uncertain databases, IEEE Trans. Knowl. Data Eng. 25 (4) (2013) 945–960.
[34] M.F. Mokbel, C.Y. Chow, W.G. Aref, The new Casper: query processing for location services without compromising privacy, in: Proc. of the 32nd Int. Conf. on Very Large Data Bases, VLDB, VLDB Endowment, 2006, pp. 763–774.
[35] M.A. Soliman, I.F. Ilyas, Ranking with uncertain scores, in: Proc. of the 25th Int. Conf. on Data Engineering, ICDE, IEEE, 2009, pp. 317–328.
[36] M.A. Soliman, I.F. Ilyas, K.C.C. Chang, Top-k query processing in uncertain databases, in: Proc. of the 23rd Int. Conf. on Data Engineering, ICDE, IEEE, 2007, pp. 896–905.
[37] Y.F. Tao, R. Cheng, X.K. Xiao, W.K. Ngai, B. Kao, S. Prabhakar, Indexing multi-dimensional uncertain data with arbitrary probability density functions, in: Proc. of the 31st Int. Conf. on Very Large Data Bases, VLDB, VLDB Endowment, 2005, pp. 922–933.
[38] V.S. Tseng, C.W. Wu, P. Fournier-Viger, S.Y. Philip, Efficient algorithms for mining top-k high utility itemsets, IEEE Trans. Knowl. Data Eng. 28 (1) (2016) 54–67.
[39] A. Vlachou, C. Doulkeridis, Y. Kotidis, K. Nørvåg, Reverse top-k queries, in: Proc. of the 26th Int. Conf. on Data Engineering, ICDE, IEEE, 2010, pp. 365–376.
[40] A. Vlachou, C. Doulkeridis, Y. Kotidis, K. Nørvåg, Monochromatic and bichromatic reverse top-k queries, IEEE Trans. Knowl. Data Eng. 23 (8) (2011) 1215–1229.
[41] A. Vlachou, C. Doulkeridis, K. Nørvåg, Monitoring reverse top-k queries over mobile devices, in: Proc. of Int. Workshop on Data Engineering for Wireless and Mobile Access, MobiDE, ACM, 2011, pp. 17–24.
[42] A. Vlachou, C. Doulkeridis, K. Nørvå g, Y. Kotidis, Identifying the most influential data objects with reverse top-k queries, Proc. VLDB Endow. 3 (1–2) (2010) 364–372.
[43] A. Vlachou, C. Doulkeridis, K. Nørvå g, Y. Kotidis, Branch-and-bound algorithm for reverse top-k queries, in: Proc. of the ACM Int. Conf. on Management of Data, SIGMOD, ACM, 2013, pp. 481–492.
[44] G.Q. Xiao, K.L. Li, K.Q. Li, Reporting l most favorite objects in uncertain databases with probabilistic reverse top-k queries, in: 2015 IEEE International Conference on Data Mining Workshop, ICDMW, IEEE, 2015, pp. 1592–1599.
[45] G.Q. Xiao, K.L. Li, K.Q. Li, X. Zhou, Efficient top-(k, l) range query processing for uncertain data based on multicore architectures, Distrib. Parallel Databases 33 (3) (2015) 381–413.
[46] G.Q. Xiao, F. Wu, X. Zhou, K.Q. Li, Probabilistic top-k range query processing for uncertain databases, J. Intell. Fuzzy Syst. (2016) 1–12, Preprint.
[47] M.L. Yiu, N. Mamoulis, X.Y. Dai, Y.F. Tao, M. Vaitis, Efficient evaluation of probabilistic advanced spatial queries on existentially uncertain data, IEEE Trans. Knowl. Data Eng. 21 (1) (2009) 108–122.
[48] A.W. Yu, N. Mamoulis, H. Su, Reverse top-k search using random walk with restart, VLDB J. 7 (5) (2014) 401–412.
[49] W.J. Zhang, X.M. Lin, Y. Zhang, W. Wang, J.X. Yu, Probabilistic skyline operator over sliding windows, in: Proc. of the 25th Int. Conf. on Data Engineering, ICDE, IEEE, 2009, pp. 1060–1071.
[50] X. Zhang, J. Chomicki, On the semantics and evaluation of top-k queries in probabilistic databases, in: Proc. of the 29th Int. Conf. on Data Engineering Workshops, ICDEW, IEEE, 2008, pp. 556–563.
[51] X. Zhang, J. Chomicki, Semantics and evaluation of top-k queries in probabilistic databases, Distrib. Parallel Databases 26 (1) (2009) 67–126.
[52] Z.J. Zhang, Y. Yang, A. Tung, D. Papadias, Continuous k-means monitoring over moving objects, IEEE Trans. Knowl. Data Eng. 20 (9) (2008) 1205–1216.
[53] X. Zhou, K.L. Li, Y.T. Zhou, K.Q. Li, Adaptive processing for distributed skyline queries over uncertain data, IEEE Trans. Knowl. Data Eng. 28 (2) (2016) 371–384.
[54] X. Zhou, Y.T. Zhou, G.Q. Xiao, Y.F. Zeng, F. Zheng, Effective approach for an extended p-skyline query, J. Intell. Fuzzy Syst. (2016) 1–10, Preprint.