Data & Knowledge Engineering 76–78 (2012) 16–38

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

DuoWave: Mitigating the curse of dimensionality for uncertain data

Chunyang Ma, Rui Zhang*, Xuemin Lin, Gang Chen

College of Computer Science, Zhejiang University, China
Department of Computing and Information Systems, University of Melbourne, Australia
School of Computer Science and Engineering, The University of New South Wales, Australia

A R T I C L E I N F O

Article history:
Received 9 December 2010
Received in revised form 26 March 2012
Accepted 26 March 2012
Available online 27 April 2012

Keywords:
Index
High-dimensional
Uncertain

A B S T R A C T

The curse of dimensionality has been a vexatious obstacle in processing queries on multidimensional data. This problem is more serious with uncertain data: an uncertain object's value may spread extensively in the data space with varying probability distribution. In this paper, we attack this challenging problem and propose a technique called DuoWave for indexing uncertain multidimensional objects under a commonly used data model. We propose efficient algorithms to process range queries, the most popular filtering paradigm for many multidimensional queries on uncertain data. Extensive experiments show that DuoWave significantly outperforms state-of-the-art techniques. Moreover, DuoWave can also be exploited for a number of other query types on uncertain data.

1. Introduction

Data in many applications exhibit impreciseness and uncertainty. Consider the following three classes of sources.

Class 1 Sensors are used to monitor the environment in a forest. The data of the environment of a forest may consist of many aspects, such as the temperature, illumination intensity, time, humidity, intensity of pressure, oxygen content, dust content in the air, wind speed and so on. Sensors send the readings to a server through a wireless network. Due to the limitations of network bandwidth and battery power of sensors, it is not practical to send the readings to the server at a high frequency. The actual environment value may keep fluctuating and therefore the reported value at the server is uncertain. Measurement errors also bring uncertainty into the data.

Class 2 A set of parameters is used to describe an object's behavior. For example, an NBA (National Basketball Association) player's game statistics are shown in Table 1. Parameters such as points scored (PTS), assists (AST), and steals (STL) are used as a composite descriptor for a player's performance. Since the player's performance varies every game, the performance descriptor may be treated as uncertain. In particular, we can view a player's performance as a random variable and the parameters at different games as samples of the random variable. Many other applications such as stock price predictors, company performance indicators, etc., may contain a large number of parameters and exhibit similar characteristics.

Class 3 Privacy preservation is a serious concern in publication of personal data. Consider a popular example in the literature where hospitals release patients' medical records for research purposes. Each record contains attributes such as the patient's age, gender, height, weight, race, education level, zipcode (or postcode), as well as the type of disease. To protect the privacy of individual patient, hospitals always generalize attributes of the released records into ranges [1,2] or inject noise into the released information about the records [3,4]. Moreover, as location privacy has become increasingly a

* Corresponding author at: Department of Computing and Information Systems, University of Melbourne, Australia.
E-mail addresses: mcy@cs.zju.edu.cn (C. Ma), rui@csse.unimelb.edu.au (R. Zhang), lxue@cse.unsw.edu.au (X. Lin), cg@cs.zju.edu.cn (G. Chen).

0169-023X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2012.03.004

Table 1
An example of data of NBA players.

Name: Kobe Bryant							
Game	OFF	DEF	AST	STL	BLK	PF	PTS
1	1	4	6	4	2	2	37
2	0	3	4	0	1	5	26
3	0	2	5	1	0	1	23
4	0	2	2	1	0	3	31
5	4	7	2	1	1	5	49

concern, many locations are blurred (e.g., points are cloaked to regions) when they are published [5,6]. Therefore such published information is always uncertain.

Data in such applications can be modeled as follows. The data objects reside in a multidimensional space. An object is modeled by a probability distribution [7,8]. In the first class, for example, the environment of a forest is changing all the time, and a probability density function is used to represent the continuous probability distribution of the object being in the possible range, whereas it is natural to use a probability mass function to represent the discrete probability distribution of the data in the second class of applications.

Range queries are the most basic and important type of queries in many applications and also used commonly as a filtering paradigm for processing many types of multidimensional queries. A range query on uncertain multidimensional data finds objects whose attribute values (the parameter values aforementioned) are in a given range with a probability greater than a given threshold. For example, find players whose points are above 30 and assists are above 4 in at least 60% of the observations. In this paper, we investigate the problem of efficiently processing range queries on uncertain data, with the focus on high dimensional spaces. To further demonstrate the effectiveness of our indexing technique, we also address another important and common query type, the *similarity query* on uncertain multidimensional data, especially in high dimensional spaces. A similarity query finds objects similar to a given object based on a probabilistic metric, which we will define formally later.

The state-of-the-art techniques for indexing uncertain objects such as U-trees [8] and APLA-trees [9] use the R-tree as the underlying structure. However, the R-tree is known to suffer from the *curse of dimensionality* when indexing multidimensional data, i.e., it performs worse than a sequential scan when intrinsic dimensionality reaches about ten [10,11]. This problem is more serious with uncertain data: the possible values of uncertain objects may diffuse throughout the data space, which causes major overlaps between bounding boxes of different objects when using a spatial index structure like the R-tree to index them. The overlaps result in low pruning power and high access costs during query processing. In the aforementioned applications, data usually have dozens of attributes and the need for an index that works well in high dimensional spaces is compelling.

Motivated by the strong need to address the high-dimensionality problem in indexing uncertain data, we propose DuoWave, a technique to in*d*ex *u*ncertain *o*bjects by *wave*lets. The DuoWave index is a compression of the original data file based on histograms and wavelets. The index serves as a filter to prune unqualified objects during query processing. After the filtering phase, a small portion of the data objects remain as candidate answers. We only need to retrieve these objects from the original data file to check if they are the actual answers. Both scanning the index and retrieving candidates from the original data file are sequential disk accesses. If the index can be made *much* smaller in size than the original data file and at the same time, the index can prune *most* objects, then we may achieve less sequential access in total than scanning the original data file, and hence provide superior performance compared to a sequential scan, irrespective of dimensionality.

The challenge lies in deriving highly effective bounds to prune objects. We propose bounds proven to be tight based on our compression scheme for processing range queries. To our knowledge, this is the first study to specifically address the "dimensionality curse" problem for general range queries on uncertain data. Moreover, DuoWave is generic and can support various types of queries. We focus here on processing range and similarity queries while we also discuss how to process a number of other types of queries on uncertain data using DuoWave. Our contributions are summarized as follows.

- We propose a new indexing scheme for uncertain data based on a compression approach, called DuoWave. It integrates several compression techniques such as histograms, wavelets, and dimension selection with careful design to achieve high efficiency.
- We propose an algorithm to process range queries based on the DuoWave index. Specifically, the algorithm adopts a filter-and-refine paradigm and uses the DuoWave index to derive some bounds for filtering unqualified objects. We prove that the bounds are tight under our DuoWave compression scheme. We use a similar strategy for similarity queries and provide an efficient algorithm. The strong pruning power of the algorithms is validated in the experimental study.
- We show how DuoWave can be utilized to process a number of other types of queries.
- We perform an extensive experimental study. The results validate the effectiveness of our compression scheme and the efficiency of our algorithms. In all the experiments, our technique consistently outperforms the U-tree, APLA and sequential scan, especially when the dimensionality is high.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 presents the data model and query definitions. Section 4 describes our indexing technique, DuoWave. We provide algorithms to process range queries in Section 5 and similarity queries in Section 6. We discuss processing of other types of queries in Section 7. Section 8 presents our experimental results and Section 9 concludes the paper and presents future work.

2. Related Work

2.1. Processing queries on uncertain data without an index

Many papers address individual types of queries on uncertain or probabilistic data, such as top-k answers for traditional SQL queries on uncertain data [12], aggregate queries on uncertain data [13,14], Top$k$-PNN (Probable Nearest Neighbor) query [15] and Constrained Probabilistic Nearest-Neighbor (C-PNN) Query [16]. The above papers focus on computing query answers efficiently, but do not propose any index structure to facilitate query processing. Pei et al. [17] address probabilistic skyline queries on uncertain data, which find objects being in the skyline with probability higher than a given threshold p. The above papers focus on computing query answers efficiently, but do not propose index structures.

2.2. Indexing techniques on uncertain data

Cheng et al. [7] propose some indexing techniques for range queries on uncertain data, but only for one-dimensional space. Recently, Aggarwal et al. [18] analyze the complexity of indexing one-dimensional uncertain data for range queries and propose indexing schemes with linear or near-linear sizes and logarithmic query time. However, the results are mostly theoretical and it is unclear how to extend the schemes to multidimensional data.

The UniGrid index is proposed for a special type of high-dimensional uncertain data where dimension-wise independent probability distribution is assumed [19]. We explain the limitation of the assumption of UniGrid with an example. For ease of explanation, we assume each object has two attributes, A and B, and probability mass functions are used to represent the probability distribution of objects' attributes. Given the UniGrid model, for one example object X, the dimension-wise independent probability distribution of attributes A and B are represented as $\{Pr(X.A=1)=0.2, Pr(X.A=2)=0.8\}$ and $\{Pr(X.B=3)=0.2, Pr(X.B=5)=0.8\}$, respectively. However we also have another object Y, which is associated with a joint probability distribution $\{Pr(Y.(A,B)=(1,5))=0.2, Pr(Y.(A,B)=(2,3))=0.2, Pr(Y.(A,B)=(2,5))=0.6\}$. Note that the marginal distribution of $Y.A$ and $Y.B$ is the same as the distribution of $X.A$ and $X.B$, respectively. However the joint distribution of the attributes of Y cannot be represented by the UniGrid model. Thus, we need a more powerful model to fit into the real data. In this paper, we assume a more general model, specifically, joint probability distribution that supports arbitrary correlations on the uncertain values of the objects.

Singh et al. [20] have addressed indexing of categorical data, where an attribute's value must be drawn from a categorical domain. They propose two methods, an inverted index approach and an R-tree approach. These methods have good performance for data with categorical domains, but do not apply to infinite data domains.

In terms of addressing the general form of range queries, the U-tree [8] and APLA [9] are comparable to our technique. U-trees use minimum bounding rectangles (MBRs) to represent objects' uncertain ranges and several nested hyper-rectangles (CFBs) to approximate the probability distribution. MBRs and CFBs guide the grouping of objects and are stored in an R-tree structure. During processing of a range query, the query range is compared with both MBRs and some of the CFBs, determined by the query probability threshold. Since U-trees use R-trees as the underlying structure, a major concern is that U-trees are not scalable to high dimensionality as discussed in Section 1. Our experiments show that U-trees perform worse than sequential scan when dimensionality is more than ten. Moreover, the U-tree only addresses range queries.

The APLA technique [9] first approximates the distribution of an uncertain object using piecewise linear functions. Then the piecewise linear approximations are indexed using an R-tree or stored in a sequential file, which results in two techniques called *APLA-tree* and *APLA-sequential*, respectively. The APLA-tree index also uses the R-tree as the underlying structure, which suffers from the same problem as the U-tree does. APLA-sequential is more promising for high-dimensional uncertain data and we will compare our technique with APLA-sequential in the experimental study.

2.3. Processing queries on certain multidimensional data

Modeling methods of high-dimensional data are summarized in [21]. Many existing studies attempt to break the *curse of dimensionality* for certain data [22,23,10,24–26]. However, there is no straightforward way to adapt any of them to index uncertain data, because those techniques were designed for point data and cannot handle data objects represented by regions of possible values with a probabilistic distribution. In addition, a sparse learning machine is developed for high-dimensional data [27].

Lin [28] proposes a novel indexing structure of line segments based on compressed B^+ trees. Predictive range and KNN queries on moving objects are addressed in [29]. Moving KNN queries in multidimensional space are addressed in [30,31]. Some works address multi-granularity indexing and query processing to support web query expansion [32].

2.4. Approximating with wavelets and histograms

Our index structure employs histograms and the wavelet transform (wavelets). Wavelets have been widely used in data streams [33], time series [34], etc., but not in indexing uncertain data. Some existing studies (e.g., [35,36]) have focused on minimizing certain error metrics when using histograms and wavelets for approximation, and so that the summaries (histograms and wavelets) can be used for probabilistic data exploration, approximate query processing, etc. Those techniques provide approximate solutions instead of exact ones. In comparison, our work uses histograms and wavelets to derive bounds, which function as a kind of filter for pruning unqualified objects. Our algorithms provide exact answers instead of approximate ones. In addition, by using different types of histograms and/or wavelets only the efficiency of our technique is going to be affected (because of possibly tighter bounds). Developing histograms and wavelets with good approximation bounds is an orthogonal issue to this paper and the histograms and wavelets proposed by previous studies can be used in our technique without affecting the correctness of the answers.

3. Data model and problem definition

In this section we define range and similarity queries based on a common uncertain data model. Frequently used symbols are summarized in Table 2.

3.1. Data model

The dimensionality of the data domain space is the number of attributes of data objects, denoted as d. A *certain object* is represented by a single point $x=\langle x_1,x_2,\ldots,x_d\rangle$ in the d-dimensional domain space, associated with a probability 1.

As discussed in Section 1, an *uncertain object* X is modeled by a probability distribution. The probability distribution of each uncertain object may be continuous or discrete. Note that for most real applications, the probability of an uncertain object being outside a certain bound is too small to be possible in practice. To make such distributions manageable, most existing studies on managing uncertain databases have modeled uncertain objects with bounded ranges [7,8,37]. Therefore we follow this approach in our paper and assume that the probability distribution of each uncertain object X is a constraint distribution so that all possible values of X are within an uncertain region $X.ur=\{[X.l_1,X.u_1], [X.l_2,X.u_2],\ldots, [X.l_d,X.u_d]\}$. Then if an uncertain object is associated with a discrete probability distribution, we use a probability mass function to represent its probability distribution, denoted as *pmf*. Let $X.m$ denote the number of possible values of X. Then we have $\sum_{j=1}^{X.m} X.pmf(x[j])=1$, where $x[j]$ is the j^{th} possible value of X and $x[j]$ is inside $X.ur$. If an uncertain object has continuous probability distribution, we use a probability density function, denoted as *pdf*, to represent its probability distribution, and we have $\int_{X.ur}X.pdf(x)\mathrm{d}\,x=1$.

3.2. Problem definition

To define the range query based on the uncertain data model, we need to introduce the following definition.

Definition 1. Given a d-dimensional rectangle $R=\{[R.l_1,R.u_1],[R.l_2,R.u_2],\ldots,[R.l_d,R.u_d]\}$ and an uncertain object X, if X is associated with continuous probability distribution ($X.pdf(x)$), the probability $\mathscr{P}_R^X$ of X residing in R is

$$\mathscr{P}_R^X = \int_{R\cap X.ur}X.pdf(x)dx \tag{1}$$

If X is associated with discrete probability distribution $(X.pmf(x))$, $\mathscr{P}_R^X$ is

$$\mathscr{P}_R^X = \sum_{j=1}^{X.m}\phi(x[j],R)\cdot X.pmf(x[j]) \tag{2}$$

Table 2
Symbols.

Symbol	Meaning
X	An uncertain object.
H	The number of buckets in histograms.
S_i^X	The histogram of probabilities of X in dimension i.
R_Q	The query range of a TPR query.
P_Q	The threshold of a TPR or TPS query.
ε	The tolerance distance of a TPS query.
$\mathscr{P}_R^X$	The probability of X residing in a rectangle R.
$\mathscr{P}_S^X$	The probability of X being ε-similar to the uncertain query object Q.
$\underline{SIP(X,i)}$	The probability of X residing in the query range in dimension i.
$\overline{SIP(X,i)}$	The upper bound of probability of X residing in the query range in dimension i.
$\hat{S}_i^X$	Wavelet coefficients for X's histogram in dimension i.
$\widetilde{S_i^X}$	A bound version of $\widehat{S_i^X}$.

where

$$\phi(x[j],R)=\begin{cases}1 & \mathit{if}\ x[j]_i\in[R.l_i,R.u_i],\ \mathit{for\ all}\ i=1,2,\ldots,d;\\ 0 & \mathit{otherwise}\end{cases} \tag{3}$$

The range query on uncertain objects is defined as follows.

Definition 2. Probabilistic range (PR) query

Given a set DB of uncertain objects and a d-dimensional rectangle R_Q, find every uncertain object X in DB that has a positive probability of residing in R_Q (i.e., $\mathscr{P}_R^X>0$).

Usually, we are only interested in objects with $\mathscr{P}_R^X$ greater than a threshold, so we further define the threshold PR query as follows.

Definition 3. Threshold probabilistic range (TPR) query

Given a set DB of uncertain objects, a d-dimensional rectangle R_Q and a threshold P_Q $(0\le P_Q\le 1)$, find every uncertain object X in DB whose probability of residing in R_Q is greater than P_Q (i.e., $\mathscr{P}_R^X>P_Q$).

To define the similarity query on the uncertain data model, we need to introduce the concept of *ε-similarity* between a certain object x and an uncertain object Q as follows.

Definition 4. ε-Similarity between a certain object and an uncertain object

Given an uncertain object Q and a tolerance distance ε_i $(\varepsilon_i>0)$ for each dimension i, if Q is associated with continuous probability distribution $(Q.pdf(q))$, the probability of a certain object $x=\langle x_1,x_2,\ldots,x_d\rangle$ being ε-similar to Q is

where $$\theta(x,Q)=\int_{Q.ur}\delta(x,q)\cdot Q.pdf(q)dq \tag{4}$$

$$\delta(x,q)=\begin{cases}1 & \mathit{if}\,|q_i-x_i|\le\varepsilon_i\ \mathit{for\ all}\ i=1,2,\ldots,d;\\ 0 & \mathit{otherwise}\end{cases} \tag{5}$$

If Q is associated with discrete probability distribution $(Q.pmf(q))$, the probability of x being ε-similar to Q is

$$\theta(x,Q)=\sum_{j=1}^{Q.m}\delta(x,q[j])\cdot Q.pmf(q[j]) \tag{6}$$

where $\delta(,)$ is the same as defined in Eq. (3).

The function δ means that we only consider x similar to a value q of Q if the difference between q and x is less than a tolerance distance ε_i in every dimension i. The smaller ε_i is, the more strict the similarity definition is.

Now we can introduce the definition of ε-similarity between two uncertain objects.

Definition 5. ε-Similarity between two uncertain objects

Given an uncertain query object Q and a tolerance distance ε_i $(\varepsilon_i>0)$ for each dimension i, if an uncertain object X is associated with continuous probability distribution $(X.pdf(x))$, the probability of X being ε-similar to Q is

$$\mathscr{P}_S^X=\int_{X.ur}\theta(x,Q)\cdot X.pdf(x)dx \tag{7}$$

If X is associated with discrete probability distribution $(X.pmf(x))$, the probability of X being ε-similar to Q is

$$\mathscr{P}_S^X=\sum_{j=1}^{X.m}\theta(x[j],Q)\cdot X.pmf(x[j]) \tag{8}$$

Fig. 1 shows an example of how to compute $\mathscr{P}_S^X$ of uncertain objects in 2-dimensional space. Both Q and X have discrete probability distributions. Object Q has three possible values $q[1]$, $q[2]$ and $q[3]$ with the probabilities of 0.5, 0.1 and 0.4, respectively. Object X has two possible values $x[1]$ and $x[2]$ with the probabilities of 0.3 and 0.7, respectively. For each possible

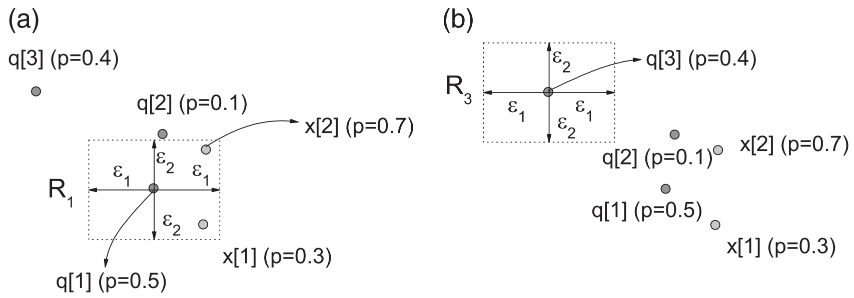

Fig. 1. ε-Similarity between two uncertain objects.

value of Q, e.g., $q[1]$, we obtain a range R_1 centered at $q[1]$ with the side length of $2\varepsilon_i$ in dimension i (c.f. Fig. 1(a)). Then we find all the possible values of X enclosed by R_1, which are $x[1]$ and $x[2]$. Similarly for $q[2]$, we find all the possible values of X enclosed by R_2 centered at $q[2]$, which is $x[2]$. For $q[3]$ (c.f. Fig. 1(b)), none of the possible values of X is in R_3. Therefore, $\mathscr{P}_S^X = 0.5 \times (0.3 + 0.7) + 0.1 \times 0.7 = 0.57$.

Now we can define the similarity query on uncertain objects.

Definition 6. Probabilistic ε-similarity (PS) query

Given a database DB, a similarity query with an uncertain query object Q and a tolerance distance ε_i ($\varepsilon_i > 0$) for each dimension i, find every uncertain object X in DB that has a positive probability of being ε-similar to Q (i.e., $\mathscr{P}_S^X > 0$).

Usually, we are also only interested in objects with $\mathscr{P}_S^X$ greater than a threshold, so we further define the threshold PS query as follows.

Definition 7. Threshold probabilistic ε-similarity (TPS) query

Given a database DB, a similarity query with an uncertain query object Q, a tolerance distance ε_i ($\varepsilon_i > 0$) for each dimension i and a threshold P_Q ($0 \le P_Q \le 1$)), find every uncertain object X in DB whose probability of being ε-similar to Q is greater than P_Q (i.e., $\mathscr{P}_S^X > P_Q$).

The PR query and the PS query are special cases of the TPR query and the TPS query, respectively. Therefore, we only present algorithms for TPR and TPS queries.

4. The DuoWave index

We describe the DuoWave index in this section. The DuoWave index is a compression of the original data file. The compression consists of three steps: (i) approximating uncertain objects' probability distributions using histograms, (ii) performing wavelet transform on the histograms, and (iii) selecting only some important dimensions to be kept in the DuoWave index. The details of these steps are given as follows.

4.1. Approximating probability distribution

We aim to accommodate data with arbitrary distribution. Accurate representation of an arbitrary distribution may take the form of a complex set of segmented functions for a continuous distribution or a large number of possible values and their probabilities for a discrete distribution. Such precise probability distribution is space consuming and we use histograms to approximate it.

When dimensionality is high, the space requirement of histograms for a multidimensional probability is prohibitive because the number of histogram buckets grows exponentially with dimensionality. Therefore, we choose to represent a projected probability distribution per dimension rather than as a joint (multidimensional) probability distribution. Note that although we approximate the probability distribution per dimension, as we will see in Section 5 and Section 6, during query processing we develop techniques to compute the information of the joint probability distribution based on these projected probability distributions. This further distinguishes our work from [19].

Two types of histogram, equi-width histogram and equi-depth histogram, can both be used here to approximate the probability distributions of uncertain objects.

If an equi-width histogram is used, for an uncertain object X with either continuous or discrete probability distribution, in each dimension, we uniformly divide the extent of $X.ur$ into H buckets and compute a histogram for the probabilities of X being in these buckets. Then we denote a histogram in dimension i by a probability sequence $S_i^X = \{p_{i,0}, p_{i,1}, \ldots, p_{i,H-1}\}$. Fig. 2 shows an example of a probability density function or a probability mass function being approximated by the histogram $S_1^X = \{0.07, 0.05, 0, 0.2, 0.1, 0.1, 0.3, 0.18\}$. The equi-width histogram is very easy to use. However when the probability distributions of objects are highly skewed, equi-width histogram may be ineffective. For example, in worst cases, one bucket may contain nearly 100% of the objects, and the sum probability of all other buckets is negligible.

An equi-depth histogram is known to work better than an equi-width histogram for highly skewed data. If an equi-depth histogram is used, for an uncertain object X with either continuous or discrete probability distribution, in each dimension we divide the extent of $X.ur$ into H buckets in a way that the probabilities of X being in these buckets are exactly the same and equal to $\frac{1}{H}$. Then we denote a histogram in dimension i by a width sequence $W_i^X = \{w_{i,0}, w_{i,1}, \ldots, w_{i,H-1}\}$, where $w_{i,j}$ is the width of the $(j+1)^{th}$ bucket.

However note that for an uncertain object X with discrete probability distribution, a pure equi-depth histogram may be not feasible, because having exactly the same "portion" of X in each bucket is impossible in some cases. Consider an uncertain object X with $X.m$ possible values. If each possible value of X has equal probability and $X.m$ is a prime number, in each dimension there is no way to partition X into H buckets so that each bucket contains exact $\frac{1}{H}$ "portion" of X, unless H is equal to 1 or $X.m$. An intuitive way to deal with such cases is to build a histogram that minimizes the probability difference between largest and smallest bucket, which are the buckets with the maximum and minimum probability of X, respectively. Let $p_{i,max}$ denote the maximum probability of such a histogram. To avoid false negatives in the answer, within DuoWave, we assume that all these H buckets contain $p_{i,max}$ "portion" of X. Then we denote the equi-depth histogram in dimension i by a width sequence W_i^X and the maximum probability $p_{i,max}$. However this way can easily cause overestimation of the probabilities since the sum probability of these buckets is larger than 1. Furthermore, the time cost to compute an equi-depth histogram is much higher than an equi-width histogram.

Both types of histograms have advantages and disadvantages. In the rest of the paper we focus on equi-width histogram for its simplicity and easy use for uncertain objects with either continuous or discrete probability distribution. For equi-depth histograms, we give brief description of how to extend the way of handling equi-width histograms to handle it and leave detailed investigation as future work.

4.2. Compression with wavelet transform

To further compress data, we perform the Haar wavelet transform on the histogram sequences and then cut off less important wavelet coefficients.

As stated above, in this subsection we mainly focus on the probability sequences (S_i^X) resulted in approximating probability distributions with equi-width histograms. We use an example to illustrate how the wavelet transform works. Full details of the wavelet transform can be found in textbooks such as [38]. Consider the probability sequence S_1^X given in the previous subsection. The first step is to average the values pairwise and obtain a sequence of half the original length: {0.06, 0.1, 0.1, 0.24}, which is a coarser-resolution version of the original sequence. The difference between the two values in each pair (the first minus the second) divided by 2 is stored as a coefficient, so we have four coefficients [0.01, −0.1, 0, 0.06]. The number of coefficients (which is the same as number of values in the resultant sequence) is called the resolution. Next, the above process is applied recursively on the resultant sequence until reaching resolution 1. Then we obtain the decomposition of S_1^X as shown in Table 3. The resolution 1 sequence (which is the average of the whole sequence) together with all the coefficients is the *Haar wavelet transform* of the original sequence. The wavelet transform (we omit "Haar" in the remainder of the paper) of S_1^X, denoted as $\widehat{S_1^X}\{\hat{c}_{1,0}, \ldots, \hat{c}_{1,7}\}$, is $\{0.125, -0.045, -0.02, -0.07, 0.01, -0.1, 0, 0.06\}$. It can be represented by a coefficient tree as shown in Fig. 3.

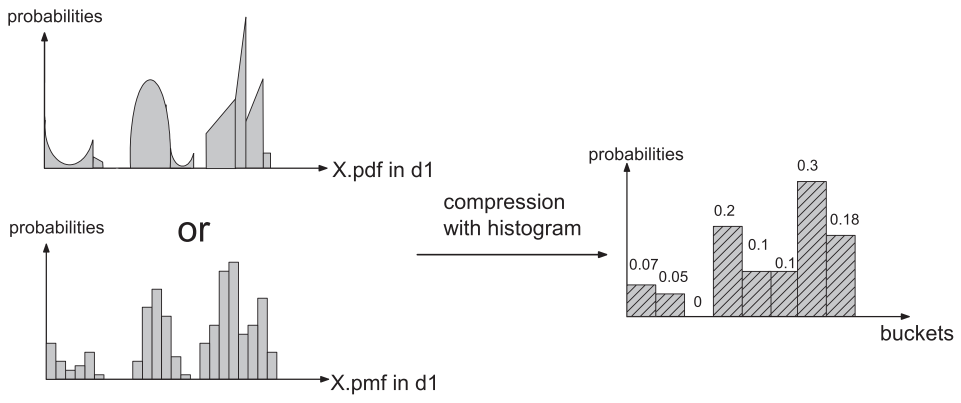

Fig. 2. Approximating probability distribution.

Table 3
Haar wavelet transform.

Resolution	Averages	Coefficients
8	{0.07 ; 0.05 ; 0 ; 0.2 ; 0.1 ; 0.1 ; 0.3 ; 0.18}	
4	{0.06, 0.1, 0.1, 0.24}	[0.01, −0.1, 0, 0.06]
2	{0.08, 0.17}	[−0.02, −0.07]
1	{0.125}	[−0.045]

The internal nodes represent coefficients and leaf nodes represent elements in the original sequence. We can derive the values of the leaf nodes from the internal nodes. A coefficient $\hat{c}_{i,m}$ only affects the values of the leaf nodes in its subtree, and we call such leaf nodes the *affected leaf nodes* of $\hat{c}_{i,m}$. For the example in Fig. 3, $p_{1,4}, p_{1,5}, p_{1,6}, p_{1,7}$ are the affected leaf nodes of $\hat{c}_{1,3}$. A coefficient $\hat{c}_{i,m}$ affects its left and right subtree in the following way. The value of $\hat{c}_{i,m}$ is added to (subtracted from) its corresponding sequence element in the same resolution to derive the coefficients in $\hat{c}_{i,m}$'s left subtree (right subtree). For the example in Fig. 3, $\hat{c}_{1,3}$, the second coefficient in resolution 2 is added to (subtracted from) 0.17, the second sequence element of resolution 2, to derive the third (fourth) sequence element of resolution 4. This relationship is described formally by the following Lemma.

Lemma 1. *Let leaves$(\hat{c}_{i,m})$ be the affected leaf nodes of a coefficient $\hat{c}_{i,m}$, LS_m (RS_m) be $\hat{c}_{i,m}$'s left (right) son. $\hat{c}_{i,m}$ affects the value of its affected leaf node $p_{i,j}$ in the following way. If the value of $\hat{c}_{i,m}$ changes to $\hat{c}'_{i,m}$ $(\Delta_m = \hat{c}'_{i,m} - \hat{c}_{i,m})$, then the value of $p_{i,j}$ changes to*

$$p'_{i,j} = \begin{cases} p_{i,j} + \Delta_m & \text{if } p_{i,j} \in leaves(LS_m) \\ p_{i,j} - \Delta_m & \text{if } p_{i,j} \in leaves(RS_m) \end{cases} \tag{9}$$

Compression based on wavelets is achieved by omitting insignificant coefficients. To determine which ones are insignificant, the coefficients are first normalized, since the coefficients in higher levels of the tree (i.e., of lower levels of resolutions) carry more information than the ones in lower levels of the tree [38]. The normalization is usually performed as follows. A coefficient at level L is divided by $\sqrt{2^L}$, with level 1 starting from the root of the tree. After normalization, $\widehat{S_1^X}$ becomes {0.088, −0.032, −0.01, −0.035, 0.004, −0.035, 0, 0.021}. The normalized coefficients with absolute values no higher than a threshold T_a are omitted. If $T_a = 0.005$, then $\hat{c}_{1,4}$ (0.004) and $\hat{c}_{1,6}$ (0) are omitted. Further, we can always omit storing $\hat{c}_{i,0}$ since it is the average of all probabilities, which can be computed as $1/H$. We use a bitmap called *coe-bitmap* to indicate omitted coefficients, with 0s in the j^{th} bit indicating the j^{th} coefficient omitted. For the above example, the coe-bitmap is 01110101. We refer to the remaining wavelet coefficients as the *abridged wavelet coefficients*.

4.2.1. Extension to equi-depth histograms

If equi-depth histograms are used, compressing the width sequences can be done in exactly the same way as compressing the probability sequences described above. In particular, we perform wavelet transform on each width sequence W_i^X and get $\widehat{W_i^X}$. Then we normalize $\widehat{W_i^X}$ and omit all coefficients in $\widehat{W_i^X}$ with absolute values no more than a threshold T_a. At last W_i^X is represented with a coe-bitmap, a threshold T_a and a set of abridged wavelet coefficients.

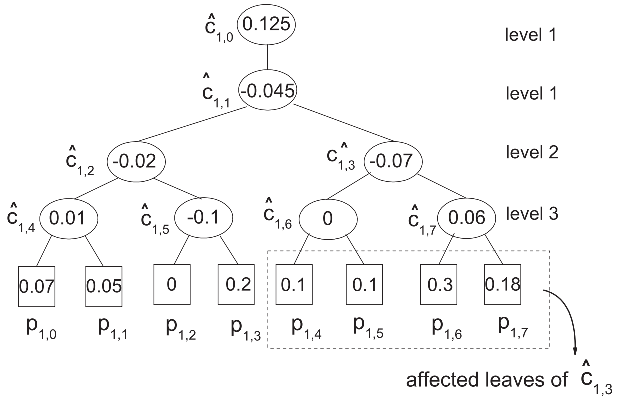

Fig. 3. An example of a coefficient tree.

4.3. Dimension selection

A dimension with no wavelet coefficients stored can be viewed as having a histogram with only one bucket and the probability of 1 in the bucket. We use a bitmap called *dim-bitmap* to indicate the abridged wavelet coefficients of which dimensions are stored. To further compress the data, we select the abridged wavelet coefficients of the most important dimensions to store. Two strategies for selecting the most important dimensions are described below.

The first strategy is to compare the extent of the uncertain region of an uncertain object in every dimension. If the extent in a dimension is small, it is not so beneficial to further partition the extent into buckets for a histogram. Therefore, the abridged wavelets in dimensions with large uncertain region extents are more important. Consider a 2-dimensional example object X, with $X.ur=\{[0.3,0.8],[0.7,0.8]\}$ and a uniform probability distribution. Given a query range R_Q, let p_1 and p_2 be the "portion" of X that intersects R_Q in dimension 1 and dimension 2, respectively. Given the same circumstances that the lengths of intersection of R_Q and $X.ur$ in both dimensions are the same, we have $p_1<p_2$ since the extent of X in dimension 2 is smaller than the extent of dimension 1. Therefore if we were to keep abridged wavelet coefficients of only one dimension, those of dimension 1 would be chosen, because approximating p_2 with 1 brings less overestimation than approximating p_1 with 1.

The second strategy is to compare the number of the abridged wavelet coefficients from each dimension. Note that one essential feature of our proposal is that the size of DuoWave should be much smaller than the original data file. Therefore, preference is given to dimensions with fewer abridged wavelet coefficients since these dimensions consume less space. In our implementation, we use the first strategy to select the dimensions and the second strategy to break ties.

4.4. Summary

Fig. 4 shows the final compressed version of a 2-dimensional example object. This compressed version is called the uncertain object's *DuoWave index entry*. The *DuoWave index* is formed by the DuoWave index entries of all the uncertain objects stored in the same order as the uncertain objects are stored in the original data file. In addition, we store some global information such as H and T_a.

4.5. Insertion, update and deletion

Inserting an uncertain object X is straightforward. We simply add the index entry of X to the end of the DuoWave file. To update an object X, we find the index entry of X and update it with the new index entry. To delete an uncertain object X, we first find the index entry of X in DuoWave, delete the index entry and mark the left space as free. To avoid too much space of DuoWave occupied by free entries, we periodically fill up the free entries by moving forward index entries behind them.

5. Processing TPR queries

In this section, we present the algorithm for processing TPR queries based on DuoWave. The algorithm follows a filter-and-refine strategy. We first scan the DuoWave index and filter out unqualified objects using upper bounds derived from the index. A small set of candidate objects remains after the filtering step and we then check these candidate objects in the original data file to determine the final answer.

A TPR query specifies a query range R_Q and a probability threshold P_Q. In the filter phase, we scan the DuoWave index entries one by one. For an uncertain object X, its index entry contains its uncertain region $X.ur$. If $X.ur$ does not intersect R_Q, X must not be an answer and we can safely discard it. If $X.ur$ intersects R_Q, we use the information in the index entry to derive an upper bound of the probability of X being in R_Q, denoted as $\overline{\mathscr{P}_R^X}$. If $\overline{\mathscr{P}_R^X}<P_Q$, X cannot be an answer and we can discard it; otherwise, X may be an answer and its ID is put in a candidate set. Then in the refine phase, we retrieve the precise values of each candidate object from the original data file and conduct the exact computation to decide which ones are really in the answer set. Note that the uncertain object IDs in the candidate set are in the same order as they are in the original data file. This makes the retrieval of the uncertain objects efficient due to sequential access.

In what follows, we discuss the most important part of the algorithm, i.e., how to use an index entry to derive an upper bound $\overline{\mathscr{P}_R^X}$ that is as tight as possible. Note that we propose a bound that is tight in its current form of one-dimensional value. It is possible to have a tight bound in the form of multi-dimensional vector on the joint probability distribution of the uncertain objects, which may have even stronger pruning power. However storing/processing joint probability distribution may bring

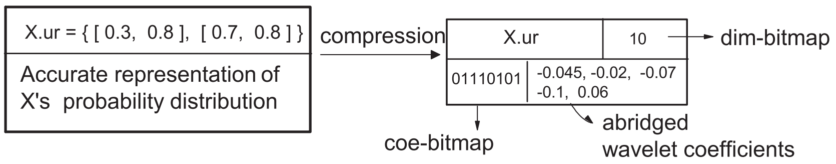
X.ur = { [0.3, 0.8], [0.7, 0.8] }
Accurate representation of X's probability distribution
compression
X.ur
10
dim-bitmap
01110101
-0.045, -0.02, -0.07
-0.1, 0.06
coe-bitmap
abridged wavelet coefficients

Fig. 4. Index entry of an uncertain object with continuous probability distribution.

considerable extra storage and CPU cost into the filter phase. There should be a trade-off between the increased CPU and I/O cost in the filter phase and the decreased cost of uncertain object accesses in the refine phase if a multi-dimensional bound is driven. We defer the full investigation of such a possible multi-dimensional bound as future work.

As stated in Section 4.1, we focus on deriving the upper bound $\overline{\mathscr{P}_R^X}$ for DuoWave that adopts equi-width histograms to approximate the objects' probability distributions, in Sections 5.1, 5.2 and 5.3. We describe a brief solution of deriving the upper bound for the approach utilizing equi-depth histogram in Section 5.4.

5.1. Deriving upper bound from histograms

When an equi-width histogram is used in DuoWave to approximate the probability distribution of uncertain objects in each dimension, we can derive an upper bound of the probability of X being in R_Q based on the histogram in each dimension as follows. For $X.ur$ to intersect R_Q, $X.ur$ must intersect R_Q in all dimensions. Let $\{p_{i,j_i}, p_{i,j_i+1} \ldots, p_{i,j'_i}\}$ denote the probabilities in the buckets intersected by R_Q in dimension i, and let $SIP(X,i)$ denote the sum of these probabilities, i.e., $SIP(X,i) = \sum_{j=j_i}^{j'_i} p_{i,j}$ for dimension i. Fig. 5 shows an example. In dimension 1, the probabilities in buckets intersected by R_Q are $\{p_{1,4}, \ldots, p_{1,7}\}$, then $SIP(X,1) = \sum_{j=4}^{7} p_{1,j}$. Similarly $SIP(X,2) = \sum_{j=3}^{7} p_{2,j}$.

Let the actual probability of X being in R_Q be $\mathscr{P}_R^X$. Then we have Lemma 2.

Lemma 2. *$min_{i=1}^{d}$ $SIP(X,i)$ is a tight upper bound of $\mathscr{P}_R^X$.*

Proof 1. First, we prove that $min_{i=1}^{d}$ $SIP(X,i)$ is an upper bound of $\mathscr{P}_R^X$, which is the percentage of possible values of X being in R_Q. For any of these possible values of X to be in R_Q, this value has to be in the extent of R_Q in every dimension. Therefore, there are at least $\mathscr{P}_R^X$ of X's possible values in R_Q in every dimension, which means that $SIP(X,i) \geq \mathscr{P}_R^X$ holds for every dimension. Therefore, $min_{i=1}^{d}$ $SIP(X,i) \geq \mathscr{P}_R^X$.

Second, we prove that $min_{i=1}^{d}$ $SIP(X,i)$ is a tight bound of $\mathscr{P}_R^X$, because there exists a case for which $\mathscr{P}_R^X = min_{i=1}^{d}$ $SIP(X,i)$. In particular, we can always construct a query R_Q and an uncertain object X that makes $\mathscr{P}_R^X$ reach the value of $min_{i=1}^{d} SIP(X,i)$ in the following way. Let i_m be the dimension that has the $min_{i=1}^{d}$ $SIP(X,i)$. Let R_Q be $\{[R.l_1, R.u_1], \ldots, [R.l_d, R.u_d]\}$. For any point x in the buckets that intersect R_Q in dimension i_m, let x satisfy $x \in R_Q$. In other words, any point in the intersected buckets of dimension i_m satisfies R_Q in all dimensions. Therefore, there is at least a $min_{i=1}^{d}$ $SIP(X,i)$ "portion" of X satisfying R_Q. The example in Fig. 5 illustrates such a query and an uncertain object, where the probability distribution of X is represented by 10 possible values, each with the same probability 0.1. In this example, $SIP(X,1) = \sum_{j=4}^{7} p_{1,j} = 0 + 0.1 + 0.3 + 0.2 = 0.6$ and $SIP(X,2) = \sum_{j=3}^{7} p_{2,j} = 0.7$. Dimension 1 has the $min_{i=1}^{d}$ $SIP(X,i)$ which is 0.6. Every point that satisfies R_Q in dimension 1 satisfies R_Q. For a query and a data uncertain object that satisfies the above description, since we know that there is at least $min_{i=1}^{d}$ $SIP(X,i)$ portion of X in R_Q, $\mathscr{P}_R^X$ is at least $min_{i=1}^{d}$ $SIP(X,i)$. On the other hand, $min_{i=1}^{d}$ $SIP(X,i) \geq \mathscr{P}_R^X$. Therefore, $\mathscr{P}_R^X = min_{i=1}^{d}$ $SIP(X,i)$.

Because there exists a case for which $\mathscr{P}_R^X = min_{i=1}^{d}$ $SIP(X,i)$ and $min_{i=1}^{d}$ $SIP(X,i)$ is no less than $\mathscr{P}_R^X$ all the time, $min_{i=1}^{d}$ $SIP(X,i)$ is a tight bound of $\mathscr{P}_R^X$.

5.2. Deriving upper bound from abridged wavelet coefficients

In DuoWave, we do not store the probability histograms but abridged wavelet coefficients on the histograms, so we can't compute the exact value of $SIP(X,i)$ of each dimension (since we do not have the histograms directly). However, based on the abridged wavelet coefficients, we can derive an upper bound of $SIP(X,i)$ for each dimension i, denoted as $\overline{SIP(X,i)}$. Then we can use $min_{i=1}^{d}$ $\overline{SIP(X,i)}$ as $\overline{\mathscr{P}_R^X}$, the upper bound of $\mathscr{P}_R^X$. Next, we show how to derive $\overline{SIP(X,i)}$ for each dimension.

If there are no wavelet coefficients stored for a dimension, $\overline{SIP(X,i)}$ of that dimension i is the largest possible value 1. Otherwise, we derive $\overline{SIP(X,i)}$ from the abridged wavelet coefficients of dimension i through the following two steps.

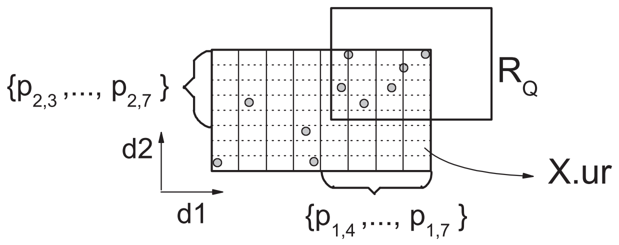

Fig. 5. A 2-dimensional example of TPR query.

Step 1 Recall that we obtain the abridged wavelet coefficients by omitting $\hat{c}_{i,0}$ and the normalized coefficients with absolute values no greater than T_a from $\widehat{S_i^X}$. The first step is to restore a bound version (denoted by $\widetilde{S_i^X}$) of $\widehat{S_i^X}$ as follows: (i) replace $\hat{c}_{i,0}$ by $1/H$ and (ii) replace other omitted coefficients by the value range $\left[-T_a\sqrt{2^L},\ T_a\sqrt{2^L}\right]$, where L is the level of the omitted coefficient in the coefficient tree (note that the threshold is applied on the normalized coefficient but we store the unnormalized coefficients). Continue the example in Section 4.2. The abridged coefficients are $\{*, -0.045, -0.02, -0.07, *, -0.1, *, 0.06\}$, where * denotes an omitted coefficient indicated by the coe-bitmap. We know $H=8$, so $\hat{c}_{1,0}=1/8=0.125$; $T_a=0.005$ and $L=3$ for $\hat{c}_{1,4}$ and $\hat{c}_{1,6}$, so $|\hat{c}_{1,4}|, |\hat{c}_{1,6}| \leq 0.005\times\sqrt{2^3}\approx 0.014$. Then $\widetilde{S_i^X}=\{0.125, -0.045, -0.02, -0.07, [-0.014, 0.014], -0.1, [-0.014, 0.014], 0.06\}$.

Step 2 The second step is to derive the upper bound of $SIP(X,i)$ from $\widetilde{S_i^X}$. From all possible $\widehat{S_i^X}$ bounded by $\widetilde{S_i^X}$, we try to find an $\widehat{S_i^X}$ corresponding to an S_i^X that maximizes $SIP(X,i)$. Towards this end, we first analyze the effect of an omitted coefficient $\hat{c}_{i,m}$ (in $\widehat{S_i^X}$) on $SIP(X,i)$. According to Lemma 1, $\hat{c}_{i,m}$ affects its affected leaf nodes in its left (right) subtree by adding (subtracting) the value of $\hat{c}_{i,m}$. Let *lcnt* (*rcnt*) be the number of probabilities that are in both $\hat{c}_{i,m}$'s left subtree (right subtree) and $\left\{p_{i,j_i}, \ldots, p_{i,j'_i}\right\}$. Then the effect of an omitted coefficient $\hat{c}_{i,m}$ (in $\hat{S}_i^X$) on $SIP(X,i)$ is as described in Lemma 3.

Lemma 3. *Given a TPR query* $<R_Q, P_Q>$, *in dimension i a coefficient* $\hat{c}_{i,m}$ *affects* $SIP(X,i)$ *in the following way. If the value of* $\hat{c}_{i,m}$ *increases by* Δ_m, *then* $SIP(X,i)$ *changes to* $SIP(X,i)'$, *where*

$$SIP\left(X,i\right)' = SIP(X,i) + \Delta_m \cdot (lcnt - rcnt) \tag{10}$$

Proof 2. Let the probabilities in $\hat{c}_{i,m}$'s affected leaf nodes be $\left\{p_{i,k_i}, p_{i,k_i+1}, \ldots, p_{i,k'_i}\right\}$. Let $LS(RS)$ denote the set of probabilities that are in both $\hat{c}_{i,m}$'s left subtree (right subtree) and $\{p_{i,j_i}, \ldots, p_{i,j'_i}\}$. Then

$$\begin{aligned} LS &= \left\{p_{i,k_i}, \ldots, p_{i,\frac{k_i+k'_i-1}{2}}\right\} \cap \left\{p_{i,j_i}, \ldots, p_{i,j'_i}\right\} \\ &= \left\{p_{i,max(k_i,j_i)}, \ldots, p_{i,min\left(\frac{k_i+k'_i-1}{2}, j'_i\right)}\right\}, \\ RS &= \left\{p_{i,\frac{k_i+k'_i+1}{2}}, \ldots, p_{i,k'_i}\right\} \cap \left\{p_{i,j_i}, \ldots, p_{i,j'_i}\right\} \\ &= \left\{p_{i,max\left(\frac{k_i+k'_i+1}{2}, j_i\right)}, \ldots, p_{i,min(k'_i,j'_i)}\right\} \end{aligned} \tag{11}$$

By our notation, $lcnt=|LS|$ and $rcnt=|RS|$. The $(lcnt+rcnt)$ elements in $LS\cup RS$ are the only probabilities in $SIP(X,i)$ that are affected by $\hat{c}_{i,m}$. Based on Lemma 1, we know that when $\hat{c}_{i,m}$ increases by Δ_m, the sum of the probabilities in LS, $\sum_{p_{i,j}\in LS} p_{i,j}$, changes to

$$\begin{aligned} \left(\sum\nolimits_{p_{i,j}\in LS} p_{i,j}\right)' &= \sum\nolimits_{j=\max(k_i,j_i)}^{\min\left(\frac{k_i+k'_i-1}{2}, j'_i\right)} p'_{i,j} = \sum\nolimits_{j=\max(k_i,j_i)}^{\min\left(\frac{k_i+k'_i-1}{2}, j'_i\right)} \left(p_{i,j}+\Delta_m\right) \\ &= \sum\nolimits_{j=\max(k_i,j_i)}^{\min\left(\frac{k_i+k'_i-1}{2}, j'_i\right)} p_{i,j} + \Delta_m \cdot lcnt = \sum\nolimits_{p_{i,j}\in LS} p_{i,j} + \Delta_m \cdot lcnt \end{aligned} \tag{12}$$

Similarly, the sum of the probabilities in RS changes to

$$\left(\sum\nolimits_{p_{i,j}\in RS} p_{i,j}\right)' = \sum\nolimits_{p_{i,j}\in RS} p_{i,j} - \Delta_m \cdot rcnt \tag{13}$$

Then the value of $SIP(X,i)$ changes to

$$SIP(X,i)' = \left(\sum_{p_{i,j}\in LS} p_{i,j}\right)' + \left(\sum_{p_{i,j}\in RS} p_{i,j}\right)' + \left(\sum_{p_{i,j}\in\left(\left\{p_{i,j_i},\ldots,p_{i,j'_i}\right\}-LS\cup RS\right)} p_{i,j}\right)' = SIP(X,i) + \Delta_m \cdot (lcnt - rcnt). \tag{14}$$

Based on Lemma 3, if $lcnt = rcnt$, the value of $\hat{c}_{i,m}$ has no effect on $SIP(X,i)$, so we can simply set $\hat{c}_{i,m}$ to 0. Otherwise, let $c_{ma} = T_a\sqrt{2^L}$ where L is the level of $\hat{c}_{i,m}$ in the coefficient tree. The minimum and maximum values of $\hat{c}_{i,m}$ are $-c_{ma}$ and c_{ma}, respectively. If $lcnt > rcnt$, $SIP(X,i)$ increases as $\hat{c}_{i,m}$ increases, so letting $\hat{c}_{i,m}$ be its maximum value c_{ma} maximizes $SIP(X,i)$. If $lcnt < rcnt$, $SIP(X,i)$ decreases as $\hat{c}_{i,m}$ increases, so letting $\hat{c}_{i,m}$ be its minimum value c_{ma} maximizes $SIP(X,i)$.

After setting all the omitted coefficients in the above way, we obtain an $\widehat{S_i^X}$ and denote it by $\widehat{S_i^X}_m$. We can restore a histogram from $\widehat{S_i^X}_m$ through the reverse wavelet transform and let us denote it by $S_i^X m$. Then we compute the sum of the probabilities in $S_i^X{}_m$'s buckets intersected by R_Q in dimension i and let us denote this sum as $SIP(X,i)_m$. Then $SIP(X,i)_m$ is used as the upper bound of $SIP(X,i)$, denoted as $\overline{SIP(X,i)}$. The detailed steps for computing $\overline{SIP(X,i)}$ are given in Algorithm 1. We can prove that:

Lemma 4. *$\overline{SIP(X,i)}$ is a tight upper bound of $SIP(X,i)$.*

Proof 3. From the derivation of $\overline{SIP(X,i)}$, we know that it upper bounds $SIP(X,i)$. Next we use a similar method used in Lemma 2 to prove that $\overline{SIP(X,i)}$ is a tight upper bound of $SIP(X,i)$.

In particular, we prove that there exists a case for which $SIP(X,i)$ can reach $\overline{SIP(X,i)}$. We can construct a query R_Q and a data object X that makes $SIP(X,i)$ reach the value of $\overline{SIP(X,i)}$ in the following way. For simplicity, we assume that the number of buckets, H, is a power of 2. Let each of the first $\frac{H}{2}$ probabilities of S_i^X be $\frac{4}{5H}$, and each of the second $\frac{H}{2}$ probabilities of S_i^X be $\frac{6}{5H}$, i.e., $p_{i,j} = \frac{4}{5H}$ for $j = 0, 1, \ldots, \frac{H}{2} - 1$ and $p_{i,j} = \frac{6}{5H}$ for $j = \frac{H}{2}, \frac{H}{2} + 1, \ldots, H - 1$, so $S_i^X = \{\frac{4}{5H}, \ldots, \frac{4}{5H}, \frac{6}{5H}, \ldots, \frac{6}{5H}\}$. Let R_Q intersect with the second $\frac{H}{2}$ buckets of X in dimension i, so R_Q intersects $\sum_{j=\frac{H}{2}}^{H-1} p_{i,j} = 0.6$ "portion" of X. Then $SIP(X,i) = 0.6$.

We perform wavelet transform on S_i^X and obtain $\widehat{S_i^X} = \{\frac{1}{H}, -\frac{1}{5H}, 0, 0, \ldots, 0\}$. Since $p_{i,1}$ becomes $-\frac{1}{5\sqrt{2}H}$ after normalization, if we choose $T_a = \frac{1}{5\sqrt{2}H}$ and cut off less important wavelet coefficients, we have no coefficients left and H 0 bits in coe-bitmap. Then we use Algorithm 1 to derive the upper bound of $SIP(X,i)$. First, we restore $\widehat{S_i^X}_m$ as follows. $\hat{c}_{i,0}$ is set to the average $\frac{1}{H}$. For $\hat{c}_{i,1}$, $c_{ma} = \frac{1}{5H}$; the number of buckets intersected with the R_Q in $\hat{c}_{i,1}$'s left subtree and right subtree are $lcnt = 0$ and $rcnt = 4$, respectively. Since $rcnt > lcnt$, we set $\hat{c}_{i,0} = -c_{ma} = -\frac{1}{5H}$. For all other coefficients, $lcnt = rcnt$, so we set all of the other coefficients to 0. Thus we have $\widehat{S_i^X}_m = \{\frac{1}{H}, -\frac{1}{5H}, 0, 0, \ldots, 0\}$. Second, we perform reverse wavelet transform on $\widehat{S_i^X}_m$ and get $S_i^X{}_m = \{\frac{4}{5H}, \ldots, \frac{4}{5H}, \frac{6}{5H}, \ldots, \frac{6}{5H}\}$, which is the same as the original histogram S_i^X. Finally, we compute the sum of the probabilities of in $S_i^X{}_m$'s buckets intersected by R_Q in dimension i, which is $\sum_{j=\frac{H}{2}}^{H-1} p_{i,j} = 0.6$, and this value is returned as the value of $\overline{SIP(X,i)}$. In the above example, $\overline{SIP(X,i)} = 0.6 = SIP(X,i)$.

Because there exists a case for which $SIP(X,i) = \overline{SIP(X,i)}$ and $\overline{SIP(X,i)}$ upper bounds $SIP(X,i)$ all the time, $\overline{SIP(X,i)}$ is a tight bound of $SIP(X,i)$.

5.3. Whole process of computing upper bound

From the abridged wavelet coefficients, we can compute $\overline{SIP(X,i)}$ for each dimension i using Algorithm 1. Then we take the minimum $\overline{SIP(X,i)}$ among all dimensions as the upper bound of $\mathscr{P}_R^X$, i.e., $min_{i=1}^d\ \overline{SIP(X,i)}$ is used as $\mathscr{P}_R^X$ in the range query algorithm to prune unqualified objects. We can prove that this bound is tight.

Theorem 1. *$min_{i=1}^d\ \overline{SIP(X,i)}$ is a tight upper bound of $\mathscr{P}_R^X$.*

Proof 4. Based on Lemma 4, $\overline{SIP(X,i)} \geq SIP(X,i)$ for every dimension i. Based on Lemma 2, $SIP(X,i) \geq \mathscr{P}_R^X$ for every dimension i. Therefore, $\overline{SIP(X,i)} \geq \mathscr{P}_R^X$ for every dimension i, and so $min_{i=1}^d\ \overline{SIP(X,i)} \geq \mathscr{P}_R^X$.

Now we prove that $min_{i=1}^d\ \overline{SIP(X,i)}$ is a tight upper bound of $\mathscr{P}_R^X$. We can always construct a query R_Q and an uncertain data object X that make $\mathscr{P}_R^X$ reach the value of $min_{i=1}^d\ \overline{SIP(X,i)}$ in the following way. Basically, R_Q and X need to satisfy the descriptions of the counter-examples constructed in Lemma 2 and Lemma 4 at the same time. Let i_m be the dimension that has the $min_{i=1}^d\ SIP(X,i)$. Let $R_Q = \{[R.l_1, R.u_1], \ldots, [R.l_d, R.u_d]\}$ intersect with the second $\frac{H}{2}$ buckets of X in dimension i_m. In dimension i_m, let the probabilities in the first $\frac{H}{2}$ buckets all be $\frac{4}{5H}$ and the probabilities in the second $\frac{H}{2}$ buckets all be $\frac{6}{5H}$. In addition, let any point x of X in the second $\frac{H}{2}$ buckets in dimension i_m be in R_Q in all dimensions. The above described R_Q and X satisfies the descriptions of the counter-examples in the proofs of both Lemma 2 and Lemma 4. Therefore, $\mathscr{P}_R^X$ reaches the bound $min_{i=1}^d\ SIP(X,i)$ which equals $SIP(X,i_m)$ and $SIP(X,i_m)$ reaches the bound $\overline{SIP(X,i_m)}$. As a result, $\mathscr{P}_R^X$ reaches the value of $min_{i=1}^d\ \overline{SIP(X,i)}$.

Since there exists a case for which $\mathscr{P}_R^X = min_{i=1}^d \overline{SIP(X,i)}$ and $min_{i=1}^d \overline{SIP(X,i)} \geq \mathscr{P}_R^X$ all the time, $min_{i=1}^d \overline{SIP(X,i)}$ is a tight bound of $\mathscr{P}_R^X$.

Algorithm 1. ComputeUPSIP(*i*, *R.l*$_i$, *R.u*$_i$)

```
if the $i^{th}$ bit in dim-bitmap is 0 then
    $\overline{SIP(X,i)} \leftarrow 1$;
else
    $j_i \leftarrow$ the first bucket intersecting with $[R.l_i, R.u_i]$;
    $j'_i \leftarrow$ the last bucket intersecting with $[R.l_i, R.u_i]$;
    //restore $\widehat{S_i^X{}_m}$ by setting its coefficients as follows
    for every coefficient $\widehat{c}_{i,j}$ of $\widehat{S_i^X{}_m}$ do
        if the $j^{th}$ bit in coe-bitmap is 0 then
            lcnt $\leftarrow$ the number of probabilities in the left subtree of $\widehat{c}_{i,j}$ and in
            $\{p_{i,j_i}, \ldots, p_{i,j'_i}\}$;
            rcnt $\leftarrow$ the number of probabilities in the right subtree of $\widehat{c}_{i,j}$ and in
            $\{p_{i,j_i}, \ldots, p_{i,j'_i}\}$;
            $L \leftarrow$ the level of $\widehat{c}_{i,j}$;
            $c_{ma} \leftarrow T_a \times \sqrt{2^L}$;
            if lcnt $=$ rcnt then
                $\widehat{c}_{i,j} \leftarrow 0$;
            else if lcnt $>$ rcnt then
                $\widehat{c}_{i,j} \leftarrow c_{ma}$ ;
            else
                $\widehat{c}_{i,j} \leftarrow -c_{ma}$;
    Perform reverse wavelet transform on $\widehat{S_i^X{}_m}$ to obtain $S_i^X{}_m$;
    $\overline{SIP(X,i)} \leftarrow$ the sum of the probabilities in $S_i^X{}_m$'s buckets intersected by $R_Q$ in
    dimension $i$;
return $\overline{SIP(X,i)}$;
```

5.4. Extension to equi-depth histogram

Recall that if equi-depth histogram is used to approximate the probability distribution of an object X, we represent the histogram of each dimension i with a width sequence W_i^X and a maximum probability $p_{i,max}$. In dimension i, let bucket j_i and bucket j'_i be the first and last buckets intersected by R_Q, $SIP(X,i) = p_{i,max} \cdot (j'_i - j_i + 1)$. Then $min_{i=1}^d SIP(X,i)$ becomes the upper bound of $\mathscr{P}_R^X$.

Since in DuoWave we do not store the width histogram but abridged wavelet coefficients on the histogram, we have to derive an upper bound of $SIP(X,i)$ (denoted as $\overline{SIP(X,i)}$) for each dimension i by deriving the lower and upper bounds for j_i and j'_i, respectively. In particular, we derive the lower (upper) bound for j_i (j'_i) through the following two steps. *Step 1*: We restore a bound version of $\widehat{W_i^X}$. *Step 2*: We choose a value for each omitted coefficient in a way that after reverse wavelet transform the value of $\sum_{j=0}^{j_i-1} w_{i,j}$ ($\sum_{j=j'_i+1}^{H-1} w_{i,j}$) is minimized and get the wavelet coefficients $\widehat{W_i^X{}_m}$. Then we restore a histogram through the reverse wavelet transform based on $\widehat{W_i^X{}_m}$ and get a width sequence $W_i^X{}_m$. Then the lower (upper) bound of j_i (j'_i) is the maximum (minimum) integer satisfying $X.l_i + \sum_{j=0}^{j_i-1} w_{i,j} \leq R.l_i$ ($X.u_i - \sum_{j=j'_i+1}^{H-1} w_{i,j} \geq R.u_i$).

After deriving the lower bound of j_i and upper bound of j'_i, we have $\overline{SIP(X,i)}$. Then we take the minimum $\overline{SIP(X,i)}$ among all dimensions as $\mathscr{P}_R^X$. Due to space limits, we leave the description of detailed algorithms and the proof of the correctness and efficiency of these algorithms as future works.

6. Processing TPS queries

In this section we present the algorithm for processing TPS queries based on the DuoWave, which still assumes a filter-and-refine paradigm. A TPS query consists of an uncertain query object $Q < Q.ur, Q.pdf(q) >$, a similarity factor ε and a threshold P_Q.

Consider the hyper-rectangle $R_{ext}=\{[Q.l_1-\varepsilon_1, Q.u_1+\varepsilon_1],\ldots,[Q.l_d-\varepsilon_d, Q.u_d+\varepsilon_d]\}$, where ε_i is the tolerance distance of dimension i. From the definition of the TPS query, it is easy to observe that an uncertain object X's uncertain region must intersect R_{ext} to have a non-zero probability of being ε-similar to Q. In the filter phase, we check the DuoWave index entries one by one. If $X.ur$ does not intersect R_{ext}, X must not be an answer and we discard it. If the $X.ur$ intersects R_{ext}, we use the information in the index entry to derive an upper bound of the probability of X being ε-similar to Q, denoted as $\overline{\mathscr{P}_S^X}$. If $\overline{\mathscr{P}_S^X}<P_Q$, X must not be an answer and we discard it; otherwise, X may be an answer and its ID is put in a candidate set. Then in the refine phase, we retrieve the precise values of each candidate object from the original data file, and conduct the exact computation to decide which ones are really in the answer set. We discuss how to use an index entry of DuoWave to derive an effective upper bound $\overline{\mathscr{P}_S^X}$ in Section 6.1.

Please note that a straightforward application of the TPR query algorithm does not work for the TPS query, because the TPS query requires that X's values not only fall in a certain range, but also follow a certain distribution; the TPR query algorithm can only help determine whether X satisfies the range requirement, but not the distribution requirement.

If the uncertain query object Q takes a discrete probability distribution, we may exploit the method of deriving the tight bound for TPR queries to derive a tight bound for the TPS query as follows. View each possible value of Q as the center of a TPR query and use $2\varepsilon_i$ as the side length of the TPR query in each dimension i, as shown in Fig. 1. The TPS query becomes a number of TPR queries, so we can derive the bounds of the probability of an uncertain data object being in each of these TPR queries, and then add these bounds up (weighted by the probability of each possible value of Q) to obtain a bound of the probability of the uncertain data object being similar to Q. However, this method is prohibitively expensive when the number of possible values of Q is large. Moreover, if Q takes a continuous probability distribution, it is impossible to use this method since there are numerous possible values of Q. Next, we propose a bound which may not be tight, but is much lighter to compute.

6.1. Deriving an upper bound

We try to take advantage of the efficient range query capability provided by the DuoWave index. The basic idea here for deriving an upper bound for the similarity query is to first relax a multidimensional query predicate (the similarity query) to one-dimensional predicates, and then each one-dimensional predicate is converted to a form that can exploit the method for deriving the upper bound of range queries. Due to the space constraint, we only present notions for objects with continuous probability distribution in this section. However the proposed approach is also adaptable for objects with discrete probability distribution.

We first show how to relax the similarity query to a number of one-dimensional predicates. Consider the following function

$$\delta_i(x,q)=\begin{cases}1 & \text{if}\,|q_i-x_i|\le\varepsilon_i\\ 0 & \text{otherwise}\end{cases} \tag{15}$$

This function is a relaxed version of $\delta(x,q)$ defined in Eq. (5); $\delta_i(x,q)$ requires that x and q be within the distance of ε_i in *only one dimension*, i.e., dimension i, instead of all dimensions. For a given point q in the uncertain region of Q, it is easy to observe that $\delta(x,q)$ represents a set of x values that is a subset of that $\delta_i(x,q)$ represents. Therefore, for Q and X with continuous probability distributions, for any dimension i ($i=1,2,\ldots,d$), we have

$$\begin{aligned}\mathscr{P}_S^X &= \int_{X.ur}\theta(x,Q)\cdot X.pdf(x)dx\\ &= \int_{Q.ur}\int_{X.ur}\delta(x,q)Q.pdf(q)X.pdf(x)dxdq\\ &\le\int_{Q.ur}\int_{X.ur}\delta_i(x,q)Q.pdf(q)X.pdf(x)dxdq\end{aligned} \tag{16}$$

Now we have relaxed the similarity probability to d one-dimensional predicates. Next, we show how to convert these one-dimensional predicates to range queries.

For dimension i, we partition the extent of Q's uncertain region into H equi-width buckets. Denote the boundaries of the f^{th} bucket as $[ql_{i,f}, qu_{i,f}]$, where $f=0,1,\ldots,H-1$. To get the bound of $\mathscr{P}_S^X$, we aggregate the similarity of X referred to each part (bucket) of Q one by one. Continue with Eq. (16), we have

$$\begin{aligned}\mathscr{P}_S^X &\le\int_{Q.ur}\int_{X.ur}\delta_i(x,q)Q.pdf(q)X.pdf(x)dxdq\\ &=\sum_{f=0}^{H-1}\int_{q\in Q.ur,q_i\in[ql_{i,f},qu_{i,f}]}\int_{X.ur}\delta_i(x,q)\cdot Q.pdf(q)X.pdf(x)dxdq\\ &=\sum_{f=0}^{H-1}\int_{q\in Q.ur,q_i\in[ql_{i,f},qu_{i,f}]}\int_{x\in X.ur,x_i\in[ql_{i,f}-\varepsilon_i,qu_{i,f}+\varepsilon_i]}Q.pdf(q)X.pdf(x)dxdq\\ &=\sum_{f=0}^{H-1}\left(\int_{q\in Q.ur,q_i\in[ql_{i,f},qu_{i,f}]}Q.pdf(q)dq\cdot\int_{x\in X.ur,x_i\in[ql_{i,f}-\varepsilon_i,qu_{i,f}+\varepsilon_i]}X.pdf(x)dx\right)\end{aligned} \tag{17}$$

for each dimension i.

Note that in Eq. (17), we can compute $\int_{q\in Q.ur,q_i\in[ql_{i,f},qu_{i,f}]}Q.pdf(q)dq$ directly. Further observe that $\int_{x\in X.ur,x_i\in[ql_{i,f}-\varepsilon_i,qu_{i,f}+\varepsilon_i]}X.pdf(x)dx$ is actually the probability of X being in the range $[ql_{i,f}-\varepsilon_i, qu_{i,f}+\varepsilon_i]$; we can use the method described in Section 5.2 to derive an upper bound for this probability from the DuoWave index of X. Then we can obtain an upper bound for $\mathscr{P}_S^X$ in every dimension i, and the smallest one among them is used as $\overline{\mathscr{P}_S^X}$, the upper bound for filtering in our algorithm. Details of the algorithm for computing $\overline{\mathscr{P}_S^X}$ are given in Algorithm 2.

Algorithm 2. ComputeSPUB(Q, ε, X)

```
$\overline{\mathscr{P}_S^X} \leftarrow 1$;
for every dimension $i$ do
    $\overline{\mathscr{P}_S^X}_i \leftarrow 0$;
    Divide $[Q.l_i, Q.u_i]$ into $H$ intervals;
    Compute $S_i^Q = \{p_{i,0}^Q, \ldots, p_{i,H-1}^Q\}$;
    for every bucket $f$ do
        if $[X.l_i, X.u_i]$ intersects with $[ql_{i,f} - \varepsilon_i, qu_{i,f} + \varepsilon_i]$ then
            $p_{i,f}^Q \leftarrow$ the sum of probability of $Q$ in bucket $f$ in dimension $i$;
            $\overline{\mathscr{P}_S^X}_i \leftarrow \overline{\mathscr{P}_S^X}_i + p_{i,f}^Q \cdot$ ComputeUPSIP($i, ql_{i,f} - \varepsilon_i, qu_{i,f} + \varepsilon_i$);
    if $\overline{\mathscr{P}_S^X} > \overline{\mathscr{P}_S^X}_i$ then
        $\overline{\mathscr{P}_S^X} \leftarrow \overline{\mathscr{P}_S^X}_i$;
return $\overline{\mathscr{P}_S^X}$;
```

7. Other types of queries

DuoWave can also support a wide spectrum of query types that may use the range query as a filter. In what follows, we discuss how DuoWave can be applied to process them.

7.1. Probabilistic dominating queries

A *Probabilistic dominating (PD) query* returns the objects with positive probability to dominate a given query object. For example, given the game records of one NBA player, find the players whose performance is sometimes better than him at all aspects. Usually we are only interested in players with probability to be better than a given player higher than a threshold. Therefore we have *Threshold probabilistic dominating (TPD) queries*, which find the objects with probability to dominate a given query object higher than a threshold P_Q.

Note that the PD and TPD queries are different from the probabilistic skyline queries [17]. For an object X being in the skyline, X must not be dominated by any other object in the database. Also X is an answer of a probabilistic skyline query if it has a probability to be in the skyline higher than a threshold. Therefore if an object X is an answer of the probabilistic skyline query lies on other objects in the database. In contrast, if X is an answer of a PD or TPD query is independent of other objects. Moreover, a probabilistic skyline query does not specify certain query object, while PD and TPD queries do.

For an uncertain query object Q, the TPD query needs to find uncertain objects whose attributes are all no less than the attributes of Q with high probability. We still use the DuoWave index to compute the upper bound of the probability of each uncertain object having all attributes no less than Q. Since Q has multiple (or numerous) possible values, the upper bound of the probability needs to be computed over each possible value of Q and then to be integrated. This is similar to the TPS query processing and therefore the way to compute the upper bound of the probability for the TPD query is similar to that for the TPS query. After computing the probability upper bounds, we can use them to filter unqualified objects.

7.2. Top-k probabilistic queries

A top-k probabilistic range (or similarity, or dominating) query (i) consists of a PR (or PS, or PD, respectively) query and a specified parameter k, and (ii) returns the k uncertain objects with the highest probability to be in a given range (or to be ε-similar to a given uncertain object, or to dominate a given uncertain object, respectively). Particularly, given an uncertain query object Q and an integer k, a top-k range (or similarity, or dominating) query retrieves from the DB a set S of k uncertain objects so that for any object $X \in S$ and any object $Y \in DB - S$, $\mathscr{P}_R^X \geq \mathscr{P}_R^Y$ (or $\mathscr{P}_S^X \geq \mathscr{P}_S^Y$, or $\mathscr{P}_D^X \geq \mathscr{P}_D^Y$).

To process any type of the top-k probabilistic queries, first we still compute the upper bound of the probability of each uncertain object being in R_Q (or being similar to Q, or dominating Q) based on the DuoWave index. For any uncertain object with the upper bound of probability greater than 0 (i.e., any uncertain object with any possibility of being in the top-k answers), we put

Table 4
The parameters of DuoWave.

Parameter	Meaning	Chosen
H	The number of histogram buckets	16
ρ	The percentage of dimensions that abridged wavelet coefficients are stored in	0.75
T_a	Threshold for compressing wavelets	0.01

them into a list. Then the list of uncertain objects is visited in the original data file one by one and we can compute the actual probability of an uncertain object being in R_Q (or being similar to Q, or dominating Q). We maintain a priority queue for the top-k candidates as we go through the list. The actual probability of the k^{th} candidate can be used as a threshold to prune objects yet to check during this process. When we finish the list, the uncertain objects in the top-k candidates form the answer set.

8. Experiments

This section presents the results of our experimental study. All the experiments are run on a desktop computer with 2.66 GHz CPU and 4 GB RAM. The page size is 4096 bytes. We use both real and synthetic datasets. The real dataset contains technical statistics of NBA players extracted from www.nba.com. Due to the limited size of the available real data, we extend the dataset in both dimensionality and cardinality to fully test effects of various parameters. Details of the real data extension, synthetic dataset generation and query generation are given in Section 8.1.

We first study the behavior of DuoWave and the effects of various parameters in Section 8.2. Then we perform comparative studies on constructing overhead, query performance of the TPR query and the TPS query in Sections 8.3, 8.4 and 8.5, respectively. For query performance test we measure the pruning power (the percentage of objects visited in the refine phase), total query response time as well as the number of disk page accesses (I/O). For every experiment, we run 100 queries and the result presented is the average of 100 queries.

Note that during all experiments, the DuoWave resides on disk and no buffer is used for it. In addition, during all experiments we deactivate the operating system buffering to achieve more rigorous test results.

8.1. Generating datasets and queries

8.1.1. Real data

The real dataset is directly extracted from the web site: www.nba.com. It contains up to 480,000 records about 1728 NBA players from 1991 to 2010. Each record has 8 dimensions including features such as points, rebounds, assists, etc. Each player is viewed as an uncertain object and the records of players are viewed as the possible values of the object. We derive the uncertain region and probability distribution for each player (object X) as follows. Suppose X has m record points. Each record is viewed as a

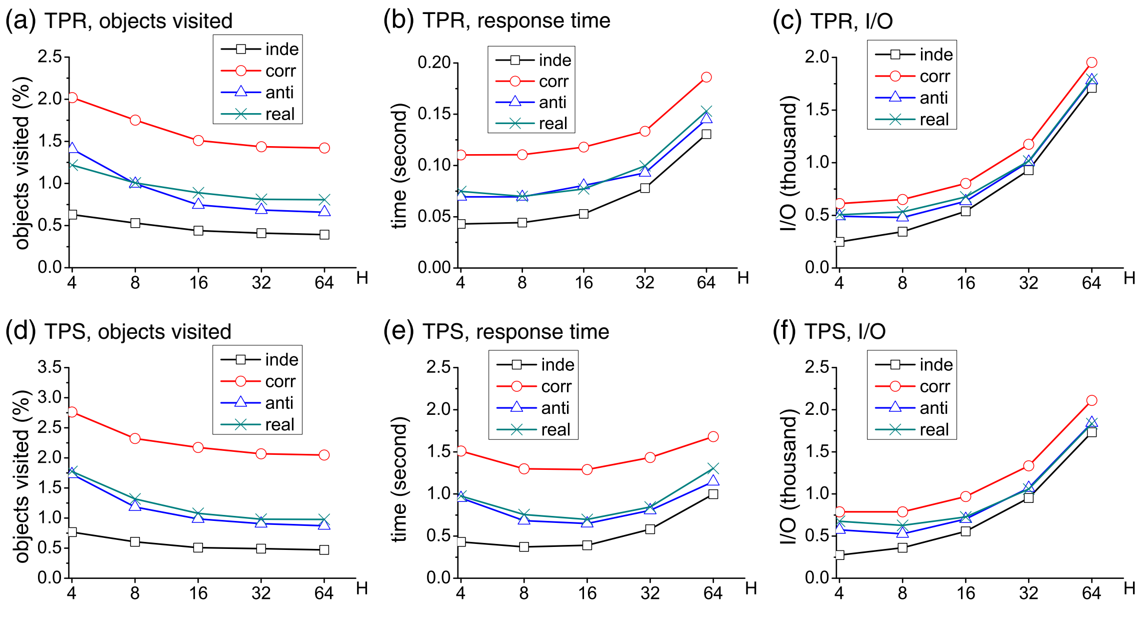

Fig. 6. Effect of H.

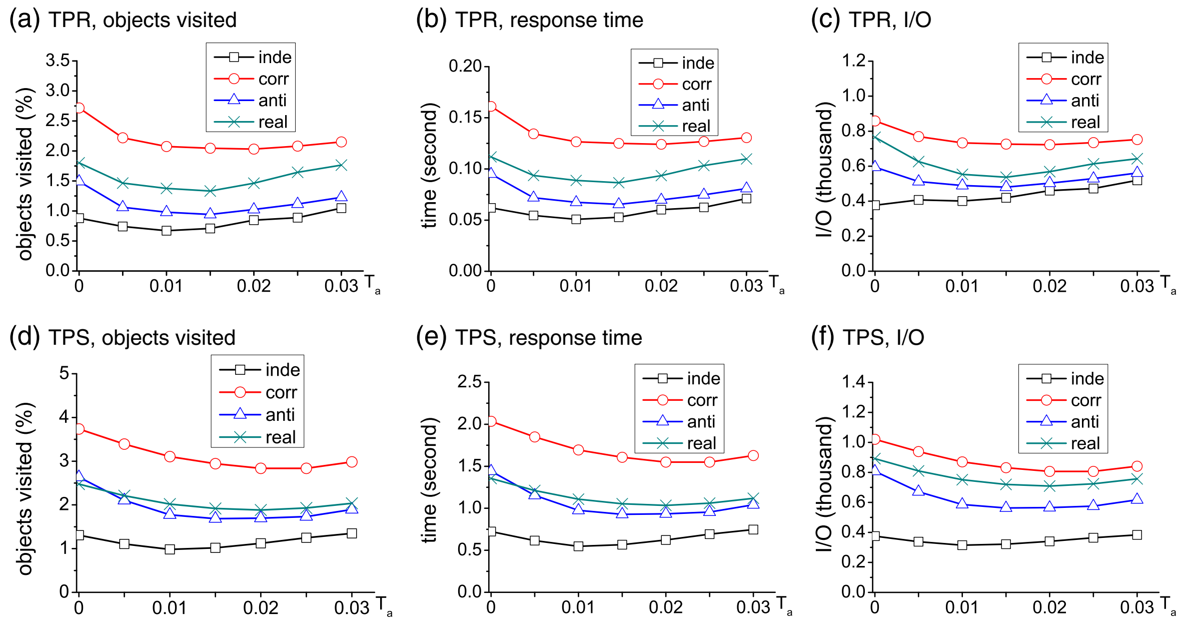

Fig. 7. Effect of T_a and ρ.

multi-dimensional possible point. Therefore $X.ur$ is the minimum hyper-rectangle that includes all these points. The probability distribution of X is represented as m possible points, each associated with a probability $\frac{1}{m}$.

In order to fully test the effects of various parameters, we have extended the dataset to larger numbers of objects and higher dimensionality in the following way. For example, given a player with records of 8 dimensions, to extend dimensionality from 8 to 16, the value of the $(i+8)^{th}$ dimension is generated by multiplying the value of the i^{th} dimension of the original records by a constant β_i $(0<i\leq 8)$, where β_i follows a constrained Gaussian distribution within $[0.8, 1.2]$. Note that β_i is used to slightly vibrate the generated values from the original values. Then we derive the uncertain region and probability distribution in the same way as described above. To generate more objects, say a new object X' based on an existing object (player) X, we use a similar idea to add some variance to the attribute values of X's records to obtain the records of X' and then derive $X'.ur$.

8.1.2. Synthetic data

We use three synthetic datasets in independent, correlated and anti-correlated distributions in our experiments. Particularly, we first generate the center points of uncertain objects using the benchmark data generator described in [39]. Then for each uncertain object X, we generate 300 points, which follow a normal distribution around its corresponding center point. Using these points, we derive the uncertain region and probability distribution of each uncertain object X in the same way as described above.

8.1.3. Queries

For TPR queries, the center points of query rectangles R_Q of TPR queries follow the same distribution of the dataset itself. The probability threshold P_Q is determined based on the average size of the TPRs to produce a desired average selectivity. For TPS queries, an uncertain query object is an uncertain object, randomly chosen from the database; ε is a user-specified parameter. The probability threshold P_Q is determined based on the uncertain query object and ε to produce a desired average selectivity in our experiments.

8.2. Effects of parameters

We first study the behavior of the DuoWave technique by testing the effects of various parameters on its performance. In this set of experiments, the dimensionality of datasets, selectivity of queries and dataset cardinality are 8, 0.1% and 10,000, respectively. Table 4 summarizes the values we finally choose for each parameter after studying the behavior of them. These parameter values are used in the comparative studies in Sections 8.4 and 8.5.

8.2.1. Effect of H

We vary H from 4 to 64 and test both kinds of queries on real and synthetic datasets. The values of ρ and T_a are chosen as 1 and 0, respectively. The test results of the real dataset and all three synthetic datasets are shown in Fig. 6.

For both types of queries on all datasets the improvement of pruning power of DuoWave is enhanced as H increases and then flattens after H goes beyond 16 as shown in Fig. 6(a) and (d).

As shown in Fig. 6(b) and (c), for TPR queries, the total response time and the number of I/Os of all the datasets increase as H increases. This is because using a larger H means a larger DuoWave index, which leads increased I/O cost. For most common probability distributions such as continuous probability distribution expressed using a closed-form function, or discrete probability distribution expressed using a small number of possible values, the computational cost of determining whether an object is an answer of a TPR query is small. Even though DuoWave with a large H can prune more objects, the effect of the stronger pruning power is outweighed by the large I/O cost caused by a large DuoWave index. For TPS queries, the computational cost of determining whether an object is an answer is higher, so the pruning power is relatively more important. This is why the response time of TPS queries on all datasets initially decreases because of the increase of pruning power, as shown in Fig. 6(e). However, when H gets too large, the effect of the stronger pruning power is still outweighed by the larger I/O cost caused by a larger DuoWave index, and hence the query performance deteriorates after H gets too large (beyond 16 in our experiments). In our experimental study, we empirically choose 16 as the default value of H for all the remaining experiments based on the results of the above experiments.

8.2.2. Effect of T_a and ρ

The values of T_a and ρ both directly affect the size of the DuoWave index and the pruning power of the DuoWave index. The DuoWave index needs to be kept small to retain a quick response time. When we vary T_a, we keep the size of DuoWave at a small portion (5% in our implementation) of the dataset size by tuning ρ at the same time. The value of T_a affects the number of abridged wavelet coefficients for each histogram and ρ determines the number of histograms in which the abridged wavelet coefficients are stored in the DuoWave index. Increasing T_a and decreasing ρ will both decrease the total number of abridged wavelet coefficients and decrease the size of DuoWave index.

As shown in Fig. 7(a) and (d), for both types of queries when T_a (and ρ) increases, the percentage of objects visited initially decreases and then increases. Overall, the change is small. Note that increasing T_a means less wavelet coefficients will be stored in DuoWave for the histogram corresponding to each dimension i, which leads to a less tighter probability upper bound of $SIP(X,i)$ for that dimension. Therefore given a fixed ρ, a larger T_a leads to a less tighter overall probability upper bound and weaker pruning power according to our compression scheme. On the other hand, to keep a fixed size of DuoWave, when T_a gets larger, ρ also becomes larger. A larger ρ means the upper bound of $SIP(X,i)$ of more dimensions will be computed, which leads to a tighter overall probability upper bound and stronger pruning power when T_a is fixed. There is a tradeoff between the effect of T_a and ρ on the pruning power of DuoWave. When increasing T_a and ρ together to keep a fixed size of DuoWave, the pruning power varies very little. Both types of queries considered the best pruning power appears when T_a is in the range [0.01,0.015] and the corresponding ρ is in ρ.

Fig. 7(b), (c), (e) and (f) shows that the change of the response time and the number of I/Os are both similar to the pruning power. The overall performance is relatively stable as long as we keep DuoWave at a fixed size. Both types of queries considered, we use 0.01 and 0.75 as the chosen values of T_a and ρ, respectively in all the remaining experiments.

8.2.3. Summary

To choose parameters of DuoWave, users can start with the parameter H. For datasets in which objects are mostly associated with continuous probability distribution functions expressed using a closed-form function or discrete probability distribution function expressed using a small number of possible values, a small size of DuoWave is preferred, so H can be set to values 8 or 16. Otherwise, users should choose a relatively larger value for H, such as 16 or 32. For cases that users do not know the details about the datasets, 16 is the recommended value for H.

After setting value for H, the best performance of DuoWave appears when the values of T_a and ρ are around $\frac{1}{5 \cdot H}$ and T_a, respectively. Setting T_a and ρ as $\frac{1}{5 \cdot H}$ and 0.75 respectively will make DuoWave provide reasonably good performance.

8.3. Comparison on insertion, update and deletion time

We compare DuoWave with the U-tree and APLA using the real dataset. For each technique, we first insert 30,000 objects. The insertion time cost is plotted as a function of the number of objects inserted as shown in Fig. 8(a). Then we update all these 30,000

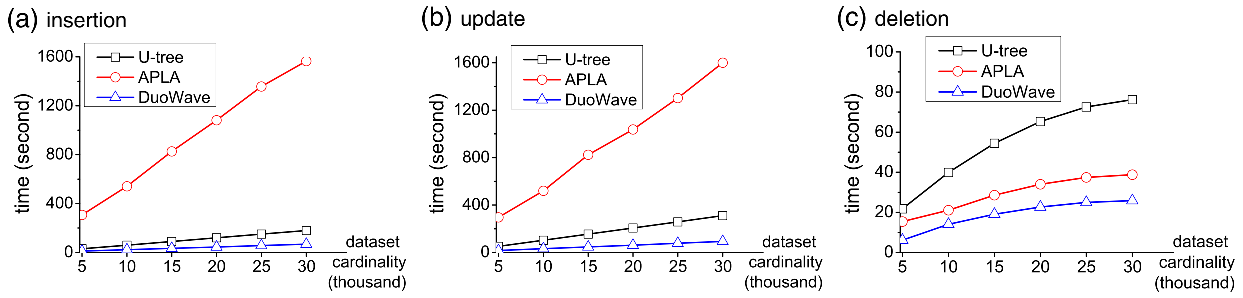

Fig. 8. Time, real dataset.

objects in a random order. The update time cost is shown in Fig. 8(b). Last, the time cost to delete all objects from each index structure in a random order is shown in Fig. 8(c).As we can see from the results, DuoWave requires the least time to insert, update and delete objects among the three indexes. Specially, compared to APLA, the time overhead of insertion and update of DuoWave is even negligible.

8.4. Comparison on TPR queries

For TPR queries, we compare DuoWave with the U-tree, the APLA technique and sequential scan. Specifically, we choose to compare with APLA-sequential rather than APLA-tree since the R-tree based structure scales badly with dimensionality. Note that except sequential scan, query processing of all methods in Fig. 9 consists of two steps, which are (i) pruning unqualified objects to obtain a candidate set with the index structure and (ii) retrieving objects from the data file sequentially. For APLA-sequential, if the parameters of APLA are set as the same as in the experiments in [9], the size of APLA index is about 20% of the dataset size. Even if it prunes abundant objects during the first step, the total cost of time and I/Os of APLA are still not good. To make the comparison fair, we make APLA with almost the same size as DuoWave by (i) letting APLA use 2 line segments (originally 3 line segments in [9]) (ii) using `float` form to store the numbers (originally `double` form in [9]) and (iii) storing APLA of 75% dimensions in APLA-sequential file. The parameters of DuoWave are set to their chosen values as shown in Table 4. The default dataset cardinality is 10,000 for all datasets. In this set of experiments, we use the real dataset and all three synthetic datasets.

8.4.1. Dimensionality

We vary the dimensionality of the dataset from 8 to 64. The selectivity of the query is 0.1%. Results are given in Fig. 9. Due to the constraint of space, the results of the anti-correlated synthetic dataset are omitted in the figure.

As shown by the results, the pruning power of all index techniques decreases as dimensionality of dataset increases. The pruning power of DuoWave is relatively stable compared with the other two indexes and DuoWave prunes the most objects

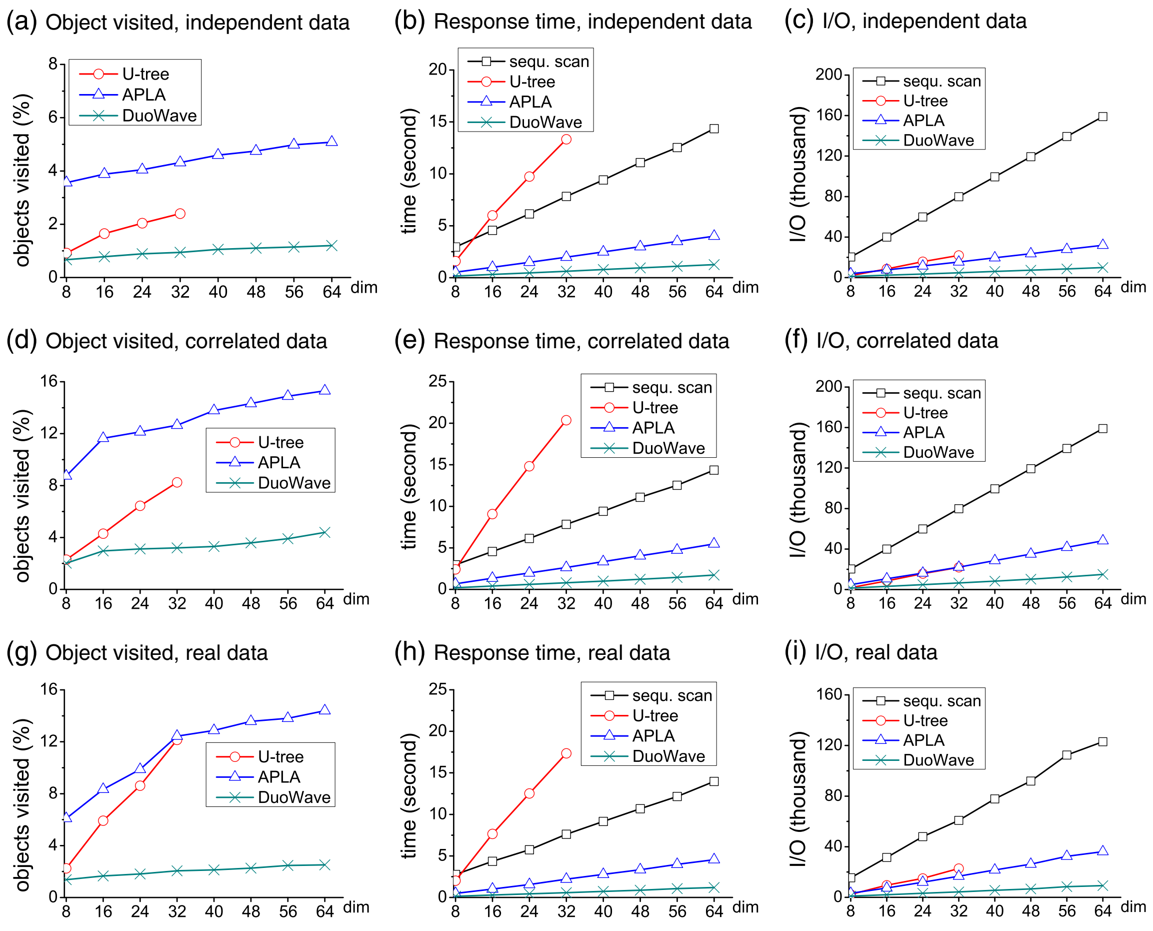

Fig. 9. TPR queries, effect of dimensionality.

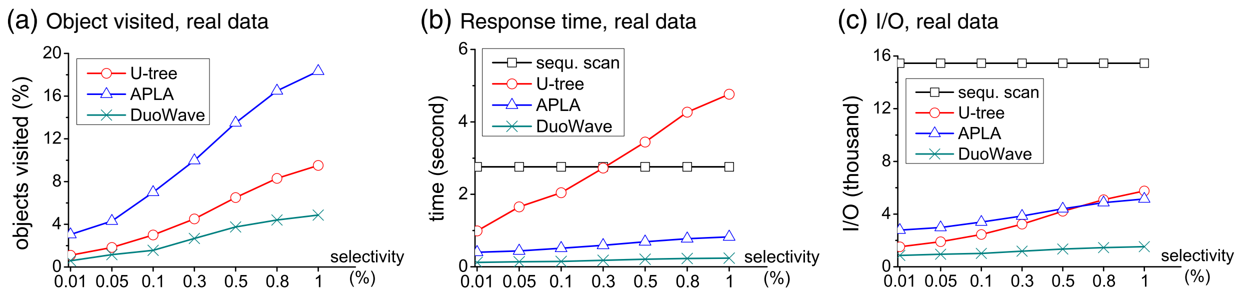

Fig. 10. TPR queries, effect of selectivity.

among all three index techniques. Particularly given the same amount of space, the pruning power of DuoWave is much better than APLA. DuoWave outperforms all the other techniques in total response time and needs the least numbers of I/Os on all datasets. For DuoWave, pruning unqualified objects is performed by sequentially scanning the pages of DuoWave. Since the size of DuoWave is only about 5% of dataset size, the time cost and the number of I/Os of pruning unqualified objects are both very low. DuoWave can always prune more than about 94% of the objects and the pruning power is stable as dimensionality grows, which means the cost of retrieving candidate objects from data file is quite small. This explains the almost linear growth in total number of I/Os and response time as dimensionality grows.

APLA mostly accesses the data sequentially. However, the pruning power of APLA is much worse. This explains why DuoWave performs better than APLA in both total response time and number of I/Os.

The curves for the U-tree stops at 32 dimensions because the U-tree implementation obtained from the original authors can accommodate datasets of at most 32 dimensions. The U-tree's performance deteriorates dramatically as dimensionality grows. We observe up to 80% of the internal nodes of the U-tree visited during the search at worst. The speedup of DuoWave over the U-tree is up to 30 times (32 dimensional real data). The most costly part of the U-tree method is traversing the tree structure which incurs considerable number of random accesses. Fig. 9(c), (f) and (i) shows that the U-tree has about 1/4 to 1/3 of the I/Os of sequential scan. Because a random access is much slower than a sequential access (usually considered at least 10 times slower), the total response time of the U-tree is much larger than sequential scan.

DuoWave consistently beats sequential scan in all the tests, irrespective of the dimensionality.

8.4.2. Query selectivity

We vary the selectivity of the TPR queries from 0.01% to 1% by changing the value of P_Q. Due to the limit of space, we only present the results of the real dataset in Fig. 10.

DuoWave outperforms other techniques significantly and always beats sequential scan, even when the selectivity reaches 1%, in which case a large number of objects are visited in the refine phase.

The performance of APLA decreases as the selectivity of TPR queries increases. Based on the results when the selectivity reaches 1%, on real dataset APLA needs to retrieve up to 20% objects in the refine phase. The response time of APLA grows evidently as the selectivity increases.

As shown by the results, when the selectivity reaches 1%, the number of I/Os of the U-tree is up to 47% that of sequential scan, which leads to a much worse response time than sequential scan due to random accesses.

8.4.3. Dataset cardinality

We vary the cardinality from 10,000 to 200,000 objects for all four datasets. The total datasets use up to 34 GB disk space. Due to limit of space we only present the results of the real dataset in Fig. 11. Again, as shown by the results on all datasets, the comparative order of the techniques remains the same. DuoWave scales very well as dataset cardinality increases.

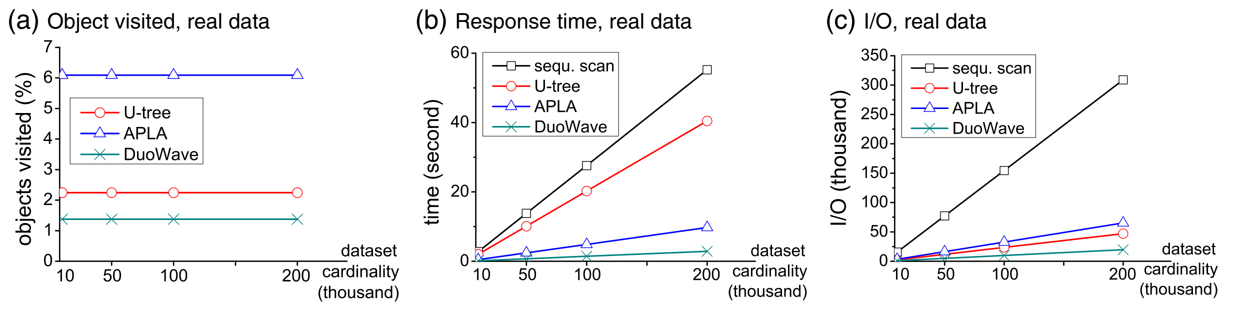

Fig. 11. TPR queries, effect of dataset cardinality.

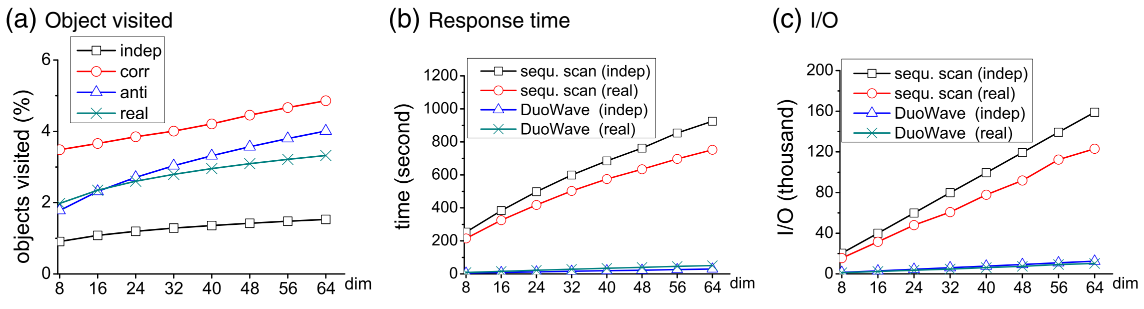

Fig. 12. TPS queries, effect of dimensionality.

8.5. Comparison on TPS queries

No previous technique addresses the TPS query. We compare DuoWave with the only obvious competitor, sequential scan. The parameters of DuoWave are set to their chosen values as shown in Table 4. The default dataset cardinality is 10,000 for both real data and synthetic data. The value of ε_i is set to be 20% of the extent of the space domain for each dimension i.

8.5.1. Dimensionality

We vary the dimensionality from 8 to 64. Fig. 12(a) shows the number of objects visited in the refine stage of DuoWave for the TPS queries on all four datasets. As shown in the figure, the pruning power of DuoWave slightly increases as the dimensionality increases. Fig. 12(b) and (c) shows the response time and the number of I/Os of the independent synthetic dataset and the real NBA dataset. The results of the other two datasets are omitted due the limit of space. As shown by the results, for all four datasets, DuoWave is more than 10 times faster than sequential scan for all dimensionalities, and the performance scales very well as dimensionality grows. Sequential scan has more I/Os on the synthetic datasets than on the real dataset because the synthetic datasets have larger file size than the real dataset.

8.5.2. Query selectivity

We vary the selectivity of the TPS queries from 0.01% to 1% by changing the value of P_Q. The comparative results between DuoWave and sequential scan are shown in Fig. 13. On all four datasets, the numbers of objects visited in the refine stage, the response time and the numbers of I/Os of DuoWave increase as the selectivity increases. However, even when the selectivity reaches 1%, the numbers of I/Os of DuoWave are still less than 10% those of sequential scan; DuoWave is still 9 times faster than sequential scan in all tests.

8.5.3. Dataset cardinality

We vary the cardinality from 10,000 to 200,000 objects and Fig. 14 shows most of the results. DuoWave always outperforms sequential scan and the performance scales well with dataset cardinality.

9. Conclusion

In this paper, we have addressed the problem of the curse of dimensionality for uncertain data. Specifically, we proposed an indexing technique called DuoWave for uncertain objects. We provide novel, efficient, DuoWave-based algorithms to process probabilistic range queries and similarity queries. We have also shown that DuoWave can support a wide spectrum of other query types that may use the range query as a filter. Extensive experiments using both synthetic and real datasets show that DuoWave significantly outperforms state-of-the-art techniques, especially in high-dimensional spaces.

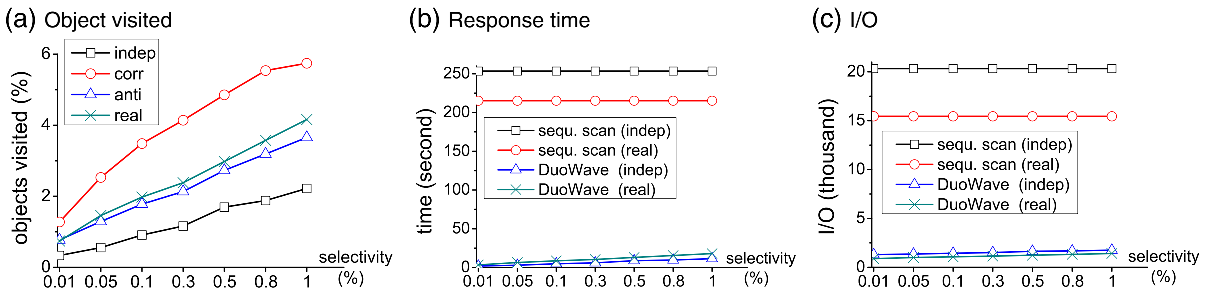

Fig. 13. TPS queries, effect of selectivity.

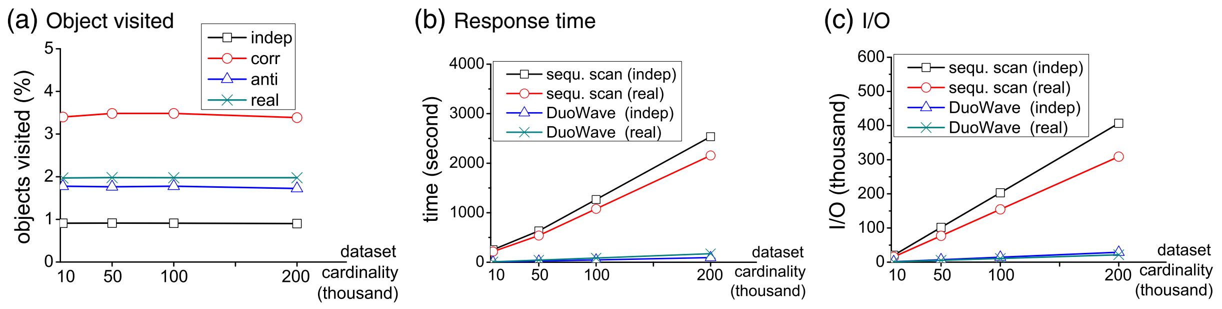

Fig. 14. TPS queries, effect of dataset cardinality.

There are several directions for future work. First we will investigate in detail the algorithms using equi-depth histograms and optimizations based on equi-depth histograms. Second, we will explore how to use our index to accommodate other types of queries on uncertain objects, introduced in Section 7. Third, we will further investigate whether an even tighter bound exists, e.g., in the form of a multi-dimensional vector on the joint probability distribution, or when certain knowledge on the data is available.

Acknowledgment

We would like to thank Alistair Moffat for his many feedbacks that have helped us improve the paper significantly. This work is supported under the Discovery funding scheme of the Australian Research Council (project number DP0880250). This work is supported in part by the National Science Foundation of China (NSFC Grant Nos. 60970124 and 61170034).

References

[1] X. Xiao, Y. Tao, Anatomy: simple and effective privacy preservation, VLDB, 2006, pp. 139–150.
[2] C.C. Aggarwal, On k-anonymity and the curse of dimensionality, VLDB, 2005, pp. 901–909.
[3] C. Dwork, Differential privacy, ICALP, 2, 2006, pp. 1–12.
[4] X. Xiao, G. Wang, J. Gehrke, Differential privacy via wavelet transforms, IEEE Transactions on Knowledge and Data Engineering 23 (8) (2011) 1200–1214.
[5] B. Bamba, L. Liu, P. Pesti, T. Wang, Supporting anonymous location queries in mobile environments with privacygrid, WWW, 2008, pp. 237–246.
[6] M.F. Mokbel, C.-Y. Chow, W.G. Aref, The new casper: query processing for location services without compromising privacy, VLDB, 2006, pp. 763–774.
[7] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, J.S. Vitter, Efficient indexing methods for probabilistic threshold queries over uncertain data, VLDB, 2004, pp. 876–887.
[8] Y. Tao, R. Cheng, X. Xiao, W.K. Ngai, B. Kao, S. Prabhakar, Indexing multi-dimensional uncertain data with arbitrary probability density functions, VLDB, 2005, pp. 922–9337.
[9] V. Ljosa, A.K. Singh, APLA: indexing arbitrary probability distributions, ICDE, 2007, pp. 946–955.
[10] R. Weber, H.-J. Schek, S. Blott, A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces, VLDB, 1998, pp. 194–205.
[11] E. Chávez, G. Navarro, R.A. Baeza-Yates, J.L. Marroquín, Searching in metric spaces, ACM Computing Surveys 33 (3) (2001) 273–321.
[12] C. Re, N.N. Dalvi, D. Suciu, Efficient top-k query evaluation on probabilistic data, ICDE, 2007, pp. 886–895.
[13] D. Burdick, P. Deshpande, T.S. Jayram, R. Ramakrishnan, S. Vaithyanathan, Olap over uncertain and imprecise data, VLDB, 2005, pp. 970–981.
[14] D. Burdick, A. Doan, R. Ramakrishnan, S. Vaithyanathan, Olap over imprecise data with domain constraints, VLDB, 2007, pp. 39–50.
[15] G. Beskales, M.A. Soliman, I.F. Ilyas, Efficient search for the top-k probable nearest neighbors in uncertain databases, PVLDB, 2008, pp. 326–339.
[16] R. Cheng, J. Chen, M.F. Mokbel, C.-Y. Chow, Probabilistic verifiers: evaluating constrained nearest-neighbor queries over uncertain data, ICDE, 2008, pp. 973–982.
[17] J. Pei, B. Jiang, X. Lin, Y. Yuan, Probabilistic skylines on uncertain data, VLDB, 2007, pp. 15–26.
[18] P.K. Agarwal, S.-W. Cheng, Y. Tao, K. Yi, Indexing uncertain data, PODS, 2009, pp. 137–146.
[19] C.C. Aggarwal, P.S. Yu, On high dimensional indexing of uncertain data, ICDE, 2008, pp. 1460–1461.
[20] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, S.E. Hambrusch, Indexing uncertain categorical data, ICDE, 2007, pp. 616–625.
[21] S.S. Vempala, Modeling high-dimensional data: technical perspective, Communications of the ACM 55 (2) (2012) 112–112.
[22] S. Berchtold, C. Bohm, H.-P. Kriegel, The pyramid-technique: towards breaking the curse of dimensionality, SIGMOD, 1998, pp. 142–153.
[23] H. Jagadish, B.C. Ooi, K.-L. Tan, C. Yu, R. Zhang, iDistance: an adaptive B^+-tree based indexing method for nearest neighbor search, TODS 30 (2) (2005) 364–397.
[24] R. Zhang, B.C. Ooi, K.-L. Tan, Making the pyramid technique robust to query types and workloads, ICDE, 2004, pp. 313–324.
[25] R. Zhang, P. Kalnis, B.C. Ooi, K.-L. Tan, Generalized multidimensional data mapping and query processing, ACM Transactions on Database Systems 30 (3) (2005) 661–697.
[26] S. Balko, I. Schmitt, G. Saake, The Active Vertice method: a performant filtering approach to high-dimensional indexing, Data Knowl. Eng. 51 (3) (2004) 369–397.
[27] Q. Cheng, A sparse learning machine for high-dimensional data with application to microarray gene analysis, IEEE/ACM Transactions on Computational Biology and Bioinformatics 7 (4) (2010) 636–646.
[28] H.-Y. Lin, Efficient and compact indexing structure for processing of spatial queries in line-based databases, Data Knowl. Eng. 64 (1) (2008) 365–380.
[29] R. Zhang, H.V. Jagadish, B.T. Dai, K. Ramamohanarao, Optimized algorithms for predictive range and KNN queries on moving objects, Information Systems 35 (8) (2010) 911–932.
[30] S. Nutanong, R. Zhang, E. Tanin, L. Kulik, Analysis and evaluation of v*-*k*nn: an efficient algorithm for moving *k*nn queries, VLDB Journal 19 (3) (2010) 307–332.
[31] S. Nutanong, R. Zhang, E. Tanin, L. Kulik, The v*-diagram: a query-dependent approach to moving knn queries, PVLDB (2008) 1095–1106.
[32] W.-S. Li, D. Agrawal, Supporting web query expansion efficiently using multi-granularity indexing and query processing, Data Knowl. Eng. 35 (3) (2000) 239–257.

[33] A.C. Gilbert, Y. Kotidis, S. Muthukrishnan, M. Strauss, Surfing wavelets on streams: one-pass summaries for approximate aggregate queries, VLDB, 2001, pp. 79–88.
[34] K. Chakrabarti, M.N. Garofalakis, R. Rastogi, K. Shim, Approximate query processing using wavelets, VLDB, 2000, pp. 111–122.
[35] G. Cormode, A. Deligiannakis, M.N. Garofalakis, A. McGregor, Probabilistic histograms for probabilistic data, PVLDB 2 (1) (2009) 526–537.
[36] G. Cormode, M.N. Garofalakis, Histograms and wavelets on probabilistic data, CoRR abs, 2008, pp. 1142–1157.
[37] R. Cheng, D.V. Kalashnikov, S. Prabhakar, Querying imprecise data in moving object environments, IEEE Trans. Knowl. Data Eng. 16 (9) (2004) 1112–1127.
[38] E.J. Stollnitz, A.D. DeRose, D.H. Salesin, Wavelets for Computer Graphics: Theory and Applications, Morgan Kaufmann, San Francisco, Calif., 1996.
[39] S. Börzsönyi, D. Kossmann, K. Stocker, The skyline operator, ICDE, 2001, pp. 421–430.

Chunyang Ma received her BSc degree in Computer Science from Zhejiang University, China, in 2006. Since 2006, she has been a PhD candidate with the College of Computer Science, Zhejiang University. From May 2008 to April 2009, she was a visiting student in the Department of Computer Science and Software Engineering, University of Melbourne. Her research interests include spatial database, spatio-temporal database and data access methods.

Rui Zhang obtained his Bachelor's degree from Tsinghua University and PhD from National University of Singapore. He has been an intern at AT&T labs-research and Microsoft Research, Redmond. He is currently a senior lecturer in the Department of Computing and Information Systems, the University of Melbourne. His research interest is data and information management in general, particularly in areas of indexing techniques, spatio-temporal databases, moving object management, web services, data streams and sequence databases.

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

Gang Chen received the PhD degree in computer science from Zhejiang University, Hangzhou, China, in 1998. Since 1998, he has been a faculty member in the College of Computer Science, Zhejiang University. From 1999 to 2003, he was an associate professor of computer science at Zhejiang University, and in 2003, he was promoted as a professor. His research interests include database management, information security, cooperative design, and CIMS.