Received June 2, 2021, accepted June 27, 2021, date of publication July 5, 2021, date of current version July 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3094547

Efficient Skyline Computation on *Uncertain Dimensions*

NURUL HUSNA MOHD SAAD[1], HAMIDAH IBRAHIM[1], (Member, IEEE), FATIMAH SIDI[1], RAZALI YAAKOB[1], AND ALI A. ALWAN[2]

[1]Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Malaysia
[2]Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, Kuala Lumpur 53100, Malaysia

Corresponding author: Hamidah Ibrahim (hamidah.ibrahim@upm.edu.my)

This work was supported by the Ministry of Higher Education Malaysia under the Fundamental Research Grant Scheme (FRGS/1/2016/ICT04/UPM/01/2) and the Universiti Putra Malaysia.

ABSTRACT The database community has observed in the past two decades, the growth of research interest in preference queries, each of which has its unique techniques, benefits, and drawbacks. One of them is skyline queries. Skyline queries aim to report to users *interesting* objects based on their preferences. Yet, they are not without their limitations. Hence, this paper focuses on efficiently extending skyline query processing to support the uncertainty in dimensions, which in this paper is defined as *uncertain dimension*. To process skyline queries on data with uncertain dimensions, we propose *SkyQUD* algorithm, where it provides a mechanism that will partition the dataset according to the characteristics of each object before skyline dominance tests are performed. In the pruning process, we utilise a probability threshold value τ to accommodate the large skyline size reported by *SkyQUD* due to the computed probabilities. The algorithm has been validated through extensive experiments. Its results exhibit that skyline queries can be performed effectively on *uncertain dimensions*, and the proposed algorithm is efficient in query answering and capable of handling large datasets.

INDEX TERMS Continuous uncertainty model, preference evaluation, skyline query, uncertain data, uncertain dimensions.

I. INTRODUCTION

In a database system, conventional SQL queries are acknowledged for having strict constraints and reporting an exact match and complete result set. Nonetheless, due to the growing number of applications that handles massive amounts of data, the huge result set reported by the conventional SQL queries has become very challenging to manage. Hence, it stirred the interest in researchers to study queries that are able to report a more concise and meaningful result set. This is where skyline query comes into existence, as it is able to capture the desires of the user to procure a result set that contains only the most interesting and preferable objects. Skyline query is able to minimize the problems faced by conventional queries by taking into considerations the preferences of the users.

Skyline queries retrieve a set of objects that are not dominated by any other objects in a dataset. Assuming that for all dimensions minimum values are more desirable, an object V is said to dominate another object W if and only if 1) V has a lower value than W in at least one dimension and 2) V has a lower value or equal to W in all other dimensions. These dominant objects are the skyline objects of the dataset, where there exists no other object in the dataset with better features. A dominance relation between objects in skyline queries is always transitive, in that if object A dominates object B, and object B in turn dominates object C, then object A dominates object C as well. A scoring function to define the relative significance of all features is not required in a skyline query, as skyline objects are selected based on the actual values of the data itself.

Consider a simple database that contains information on hotels such as room price and distance to the beach. Assume a user is interested in looking for hotels that are as cheap as possible and as close as possible to the beach. Applying skyline query on the database would retrieve all cheap hotels that are near to the beach, while having no other hotel that is cheaper and closer to the beach. Fig. 1 illustrates the results

The associate editor coordinating the review of this manuscript and approving it for publication was Nagarajan Raghavan.

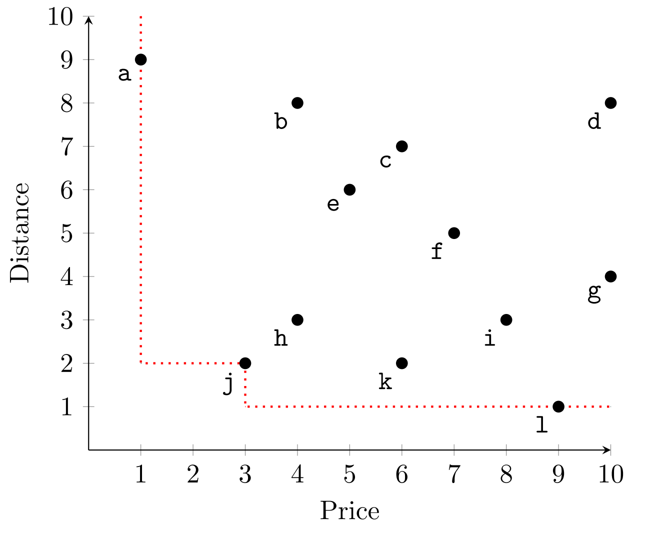
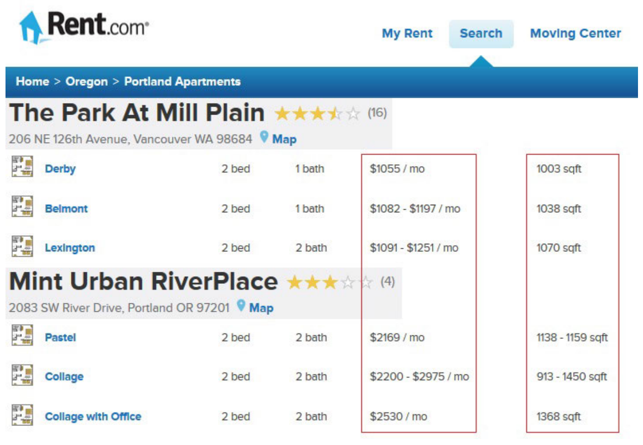


FIGURE 1. Results of skyline query for hotel database.



The Park At Mill Plain ★★★½☆ (16)
206 NE 126th Avenue, Vancouver WA 98684 Map

Derby	2 bed	1 bath	$1055 / mo	1003 sqft
Belmont	2 bed	1 bath	$1082 - $1197 / mo	1038 sqft
Lexington	2 bed	2 bath	$1091 - $1251 / mo	1070 sqft

Mint Urban RiverPlace ★★★☆☆ (4)
2083 SW River Drive, Portland OR 97201 Map

Pastel	2 bed	2 bath	$2169 / mo	1138 - 1159 sqft
Collage	2 bed	2 bath	$2200 - $2975 / mo	913 - 1450 sqft
Collage with Office	2 bed	2 bath	$2530 / mo	1368 sqft

FIGURE 2. Extraction of web data on rentals from *Rent.com*.

of performing the above skyline query on the hotel database where hotels *a*, *j*, and *l* are the results.

The skyline computation's complexity is greatly influenced by the number of dimensions and the dataset's size. The skyline queries' search space is also profoundly affected by the number of dimensions. As the number of dimensions increases, the searching space's size will extensively increase as it will become more difficult for objects to be dominated [43], [50]. Therefore, most of the existing works [7]–[10], [12], [17], [23], [34], [36], [46], [47], [49], [52] on skyline queries have attempted to solve these limitations by minimising as small as possible the searching space and reducing the skylines' processing cost. The searching space in skyline queries is typically determined by the number of pairwise comparisons performed between objects, which in turn will result in an expensive cost of processing skyline queries.

Nevertheless, various scenarios such as data modification, an approximation of missing data, collection of spatiotemporal data, and inadequate equipment for measurement in the sensor network may cause uncertainty in data. One of the many scenarios can be seen when extracting web data to perform analysis on it. In obtaining web data, uncertainty and incompleteness could not be avoided, and no matter what the cause of uncertainty is, it is essential to be able to handle the uncertainty of data. For example, a property investor wanted to get insights into Vancouver's rental market. To do the analysis, he extracted the search result for ''Vancouver'' performed on *Rent.com* web page (*www.rent.com*). For illustration, Fig. 2 shows the result of extracting web data from rental listings. Data extraction has caused uncertainty in the dimensions; for instance, the rent dimension, where some of the rent values are uncertain as they are presented in different formats. With data uncertainty, the existence of an object in a database can be interpreted in many ways. It becomes more complex to compute skyline when the values of one or more dimensions are imprecise. Having to perform skyline queries on exact values is one thing but being able to compute skyline queries on values with continuous ranges efficiently is another kettle of fish.

When dealing with uncertainty in data, it does not seem desirable to completely eradicate the uncertain values as it may lead to inaccurate or incomplete query results. Generally, there are two types of model when dealing with uncertainty in data, namely: discrete and continuous. In a discrete uncertainty model, each object has instances, each of which is associated with a probability distribution. On the contrary, in a continuous uncertainty model, all objects have an associated probability density function to capture the likelihood of possible values over a continuous range.

Various researches were done to handle skyline query processing based on the above uncertainty models. These researches were done with the assumption that either (i) uncertainty in data is caused by multiple existences of instances that represent an object [1], [6], [37], [51], or (ii) the representation of values in the form of continuous ranges in a dimension causes the uncertainty [21]. Regrettably, such assumptions do not capture well the nature of data uncertainty, wherein values in a dimension are presented as both exact values and continuous ranges. This imperfection in data is inherent and inevitable in today's real-world application.

Given a dataset, in which in a dimension (attribute), each object may be represented as an exact value and a continuous range. We termed this kind of uncertainty as *uncertain dimensions*. How can we determine the dominance relation between two objects of different representations of values? Which object shall be the skyline objects on those uncertain data? Can efficient methods be developed to compute skyline probabilities when encountering those uncertain data efficiently? To our understanding, the work in [28] is the only work that has endeavoured to solve the issue regarding the motivating example described in Fig. 2, by proposing a *BBIS* algorithm. The dominance tests are performed by comparing the objects' median values, and they have used an R*-tree structure to index their objects. Regardless of the contributions, the *BBIS* algorithm [28] is not without its shortcoming. The diminished performance of the R*-tree index structure on any dataset that contains more than five dimensions is what that has glaringly affected the performance of the *BBIS* algorithm [4], [28], [35].

To solve the above issues, in [41], we have proposed a conceptual skyline framework that could efficiently cater to data that has different representation in a same dimension, and aptly termed this scenario as *uncertain dimension*. We depicted visually the data in this scenario to demonstrate the significant of having the capability to analyse and produce skyline objects on this data and how existing works might be ad a disadvantage. We have also explored the effects of having data sets with *uncertain dimensions* in relation to the dominance relation theory. Then, in [38], we proposed an algorithm as a solution to efficiently answer skyline queries with *uncertain dimensions*. The algorithm determined skyline objects through three methods that guaranteed the probability of each object being in the final skyline results.

Following these, in this paper, we make the following contributions.

- We elaborate in detail on the skyline probabilities' concept for *uncertain dimensions* by providing additional proofs.
- We discuss in detailed on the effectiveness of existing works in supporting skyline computation when given data with *uncertain dimensions*.
- We further enhance and explicate the three methods proposed in the *SkyQUD* algorithm to reduce the searching space of skyline queries on data with *uncertain dimensions*.
- We conduct extensive experiments and evaluate the *SkyQUD* algorithm against [21], [28], [37], and include an additional performance metric (i.e. number of pairwise comparisons) to measure the performance of the algorithms.

The organisation of this paper is as follows. We review the related works in Sect. II and discuss the effectiveness of applying existing algorithms in Sect. III. In Sect. IV, the notion of the probabilistic skyline on *uncertain dimensions* is proposed. Related definitions and notations to clarify the proposed algorithm are provided. In Sect. V, the *SkyQUD* algorithm is developed to compute skyline probabilities on *uncertain dimensions*. The *SkyQUD* algorithm's performance study is reported in Sect. VI. The paper is concluded in Sect. VII.

II. RELATED WORK

In the following, we will discuss the most notable works in the area of skyline query processing that have been proposed for both types of data, namely: certain and uncertain data.

A. CERTAIN DATA

Traditionally, data that are collected and stored in databases are deterministic as most applications (e.g., employee payroll, flight ticketing, stock inventory, and firm accounting) in the old days required precise query results. Below are some of the more noteworthy algorithms introduced on processing skyline over certain data.

The skylines evolution in the context of databases can be seen from Börzsönyi *et al.*'s pioneered work [7]. They have developed the *Block-Nested-Loop* (*BNL*) and the *Divide-and-Conquer* (*D&C*) algorithms. Börzsönyi *et al.* [7] have proposed SQL syntax for skyline queries as well. *BNL* retrieves skyline objects in a brute force manner by repeatedly reading a set of objects and compares them to each other in order to eliminate the dominated objects. *D&C* on the other hand divides the dataset into several different partitions to reduce the probing area and derives the local skyline objects from each partition.

Inspired by [7], Tan *et al.* [46] then proposed two algorithms to retrieve skyline objects named *Bitmap* and *Index*. *Bitmap* works by scanning the dataset and encoding each object in bitmaps in order to determine whether an object is in the skyline, while *Index* works by organising a set of objects with *d* dimensions into *d* lists. *Index* processes the lists batch by batch according to the batch with the minimum value.

Kossmann *et al.* [23] then introduced an online skyline algorithm based on the nearest neighbour search (*NN*) using R*-tree. The algorithm works by retrieving the preliminary skyline objects while the algorithm is running simultaneously. Then, Chomicki *et al.* [9] introduced presorting into *BNL* to build a more effective algorithm, which is called *Sort-Filter-Skyline* (*SFS*). The idea of *SFS* is to retrieve skyline objects that have been sorted according to a particular sorting function. Following the work on *SFS*, Godfrey *et al.* [17] have further improved it by proposing the *Linear Elimination Sort for Skyline* (*LESS*) algorithm. In principle, *LESS* is recognised for its significant enhancement from the previous *BNL* and *SFS* algorithms as it implements the advantages of both algorithms while bypassing their disadvantages. To overcome the problem in the *NN* algorithm, Papadias *et al.* [35] have proposed the *Branch-and-Bound Skyline* (*BBS*) algorithm that is based on the sorted R-tree. The algorithm is able to retrieve skyline objects in such a progressive manner in which only a few amounts of objects are needed to be accessed. However, as the *BBS* algorithm implements R-tree, the performance of the algorithm deteriorates on dataset that involves high-dimensional spaces as it is known that the performance of R-tree is only efficient up to five dimensions [35].

Over the years, there are numerous works that have been introduced to enhance further the efficiency of existing algorithms on processing skyline query [2], [26], [42], [47], [54], [56] and expand to various domains such as semantic caching of skyline queries [5], subgraph skyline search [57], skyline computation on big data [19], and skyline on distributed datasets [24].

B. UNCERTAIN DATA

In the following, we will discuss the details of both uncertainty models and survey a broad variety of related works on skyline query processing that adopt each model.

ID	PlateNo	Make	Confidence
A	W-265	Toyota	40%
	W-519	Nissan	60%
B	F-523	Honda	50%
	F-805	Mazda	10%
	F-447	Chevy	40%
C	P-858	Subaru	30%
	P-595	Ford	70%

(a)

Possible World	Probability
$W_1 = a_1, b_1, c_1$	6%
$W_2 = a_1, b_1, c_2$	14%
$W_3 = a_1, b_2, c_1$	1.2%
$W_4 = a_1, b_2, c_2$	2.8%
$W_5 = a_1, b_3, c_1$	4.8%
$W_6 = a_1, b_3, c_2$	11.2%
$W_7 = a_2, b_1, c_1$	9%
$W_8 = a_2, b_1, c_2$	21%
$W_9 = a_2, b_2, c_1$	1.8%
$W_{10} = a_2, b_2, c_2$	4.2%
$W_{11} = a_2, b_3, c_1$	7.2%
$W_{12} = a_2, b_3, c_2$	16.8%

(b)

FIGURE 3. **Example of discrete uncertain model.**

1) DISCRETE UNCERTAINTY MODEL

In a discrete uncertainty model, objects are modelled as a probability distribution that is defined on a finite set of possible instances, each of which is associated with a probability value. For example, assume a database on the vehicle speeding trap where the images of vehicles are captured in Fig. 3. From the captured images, the system tries to identify the *PlateNo* and the *Make* of each vehicle. The images might have 'noises' on them, and thus, the vehicles can only be identified by probabilities. This has led to each vehicle having instances of possible identification with a probability of confidence. Thus, each possible combination of instances of the vehicles is considered as a possible world in the uncertain database. Note that to obtain the possible world is only feasible for objects with discrete distributions. For continuous distributions, it is unrealistic as each object will have an infinite number of possible instances.

In words, under the discrete uncertainty model, each uncertain object is modelled as a finite set of instances associated with a probability distribution. Uncertain objects in the discrete uncertainty model could not exactly be represented in one state. Instead, they have more than one possible representation, i.e. instance. For example, an uncertain object V may have n instances, $v_1, v_2, \ldots, v_n$.

Pei *et al.* [37] first pioneered the concept of probabilistic skyline on uncertain data, in which each object is represented by multiple instances and is part of the skyline answer with a certain probability. They proposed two algorithms, namely: *bottomUp* (*BU*) and *topDown* (*TD*), that would report skyline objects whose probabilities are above a threshold value p. The probability of an object to be a skyline object is computed by the summation of the probabilities of its instances that are not dominated by any other object's instances. Inspired by this work, the concept of probability theory has then been widely used in relation of skyline queries on uncertain data. Yong *et al.* [51] later followed with their work, *Best First Search* (*BFS*) algorithm, by defining the concept of skyline probability for uncertain data with maybe confidence, based on which top-*k* skyline tuples with the highest probability are identified.

Böhm *et al.* [6] then introduced τ-skyline query which applies Gaussian Mixture models on the probability density function to answer skyline query on uncertain data, and the skyline objects reported by τ-skyline would all have a minimum probability of τ. To answer τ-skyline query, they have proposed two algorithms, *Priority* and *Indexed*, which approximate the probability distribution of each object with a constant number of samples. Atallah and Qi [1] addressed the issue of computing exact skyline probabilities for all objects. They debated that requiring threshold value in probabilistic skyline computations would threshold out all uncertain objects based solely on their skyline probabilities, regardless of the skyline probabilities of their instances. Therefore, they propose an algorithm (*AQ*) that computes skyline probabilities at the instance level with no threshold value.

Having been influenced by Pei *et al.* [37], there are several emerging studies that worked on extending skyline queries on uncertain data that cover various domains such as uncertain data streams [13], [29], [33], [44], [55], distributed environment [58], and probabilistic tuples [3], [18], [22], [25], [53].

Region	Temperature (C)	Humidity (%)
r_1	24.99 - 25.05	63.20 - 64.40
r_2	24.29 - 24.31	64.00 - 65.40
r_3	27.30 - 27.80	59.90 - 68.50
r_4	28.00 - 29.00	35.00 - 40.00
r_5	17.94 - 17.97	57.97 - 58.07

FIGURE 4. **Example of continuous uncertain model.**

2) CONTINUOUS UNCERTAINTY MODEL

In the continuous uncertainty model, each object is represented as a probability density function defined on a continuous range of values (e.g., line segments, intervals). For example, assume a database reading on the *temperature* and *humidity* of regions in forests using sensor networks for the environmental monitoring programs. The researchers might have deployed a large number of sensors throughout the regions. There are a lot of factors that might affect the readings on the sensors, and since each region is equipped with hundreds of sensors, thus the sensors might record different inaccurate phenomena. To represent these readings in uncertain database (as shown in Fig. 4), each region instead will be represented by a range of values depicting the temperature and humidity of each region at a particular time.

In words, under continuous uncertainty model, each uncertain object is modelled as a continuous range of value, which is associated with a probability density function representing the possible values of the object. For example, an uncertain object V possibly may be at any point within the range $[v_{min} : v_{max}]$ where v_{min} and v_{max} are the lower-bound and the upper-bound values of object V, respectively.

Influenced by the work proposed by Pei *et al.* [37], Khalefa *et al.* [21] then introduced another interesting concept of probabilistic skyline which focuses on uncertainty in data in continuous domains where an object is represented as a continuous range in one of its dimension, which is associated with a probability density function capturing the likelihood of possible values. Thus, they have proposed a *USky* algorithm which employs a filter-refine approach in two phases. The first phase is responsible for computing dominating probability of all objects in a simple manner and filtering out objects that have probability below than a threshold value. The second phase is responsible for refining the probabilities of objects from the first phase in order to get tighter bounds. The refinement process is done iteratively for each object until the following conditions are met: 1) when the calculated upper bound probability of the object is below the threshold value, which concludes that the object is not part of the skyline query answer, or 2) the calculated lower bound probability of the object is above the threshold value and the probability of the object has an error of margin that is below the tolerance value. Nonetheless, the concept of uncertainty presented by Khalefa *et al.* [21] states that all objects are presented as a continuous range in the same dimension, while in this paper, the concept of uncertainty is presented as the existence of continuous ranges in a dimension alongside the exact values. Although theoretically an object with an exact value v can be treated as a continuous range $[v_{min} : v_{max}]$, where $v_{min} = v_{max} = v$, we explain in detail in Sect. III, on why this theory is not practical and demonstrate in Sect. VI, the disadvantage of the *USky* algorithm with extensive experiments.

Li *et al.* [28] later followed with their work where the value of an attribute for uncertain objects could be represented as exact values or intervals that conform to a probability distribution. Therefore, they have proposed a progressive algorithm, named *Branch-and-Bound Interval Skyline* (*BBIS*) which is modified from the *BBS* algorithm [35], to process the interval skyline query with an optimal cost of I/O. The algorithm employs R*-tree to index the interval objects, and this ensures that the *BBIS* algorithm performs only a single access to all nodes that may contain skyline objects. The *BBIS* algorithm works by having the immediate entry in the R*-tree to correspond to the MBR of a node at the lower level, while a leaf entry corresponds to a virtual object. This virtual object is obtained from each dimension of the median value of the interval of the actual interval object. The algorithm takes advantage of the R*-tree, which incorporates a combined optimization of area, margin, and overlap of each enclosing rectangle as compared to R-tree. Nevertheless, as with all algorithms that implement R-tree and its variant, the *BBIS* algorithm has its disadvantage when performing on high dimensionality dataset [28], [35]. We demonstrate this evidence in extensive experiments performed in Sect. VI.

Uncertainty in real-world data can transpire due to various scenarios; thus some studies have proposed algorithms to cater to different domains and environments such as Saad *et al.* [39], [40] who extended the study in [38] to report skyline on *uncertain dimensions* when given interval queries, Huang [20] who worked on the continuous d_ε-skyline query to cater to location-based query for objects with time-varying attributes, Li *et al.* [27], who extensively studied skyline query over distributed interval data, and Ma'aruf *et al.* [30]–[32], who worked on an alternative approach from [38] to cater skyline queries on uncertain data. On a different perspective from the two domains discussed previously, Elmi *et al.* [15], [16] introduced the skyline paradigm that focuses on the evidential database, Dzolkhifli *et al.* [14] worked on analysing interval uncertain data stream with k-means clustering technique, while Dehaki *et al.* [11] proposed a rule-based skyline computation for data in dynamic database.

III. DISCUSSION ON THE EXISTING ALGORITHMS

Below we discuss the three most significant existing works ([21], [28], [37]) that adopt the uncertainty model in supporting skyline computation. These works will be discussed based on the following property. Given a two 2-dimensional dataset $\mathbb{D} = (\mathbb{C}(D_1), D_2)$, where $\mathbb{C}(D_1)$ represents a dimension that contains values that are in the form of continuous range and exact values, while D_2 represent a dimension that contains values that are in the form of exact values only. Two objects $x, y \in \mathbb{D}$ have the following values, $x = ([1 : 5], 2)$ and $y = (3, 1)$.

Pei *et al.* [37] proposed *probabilistic skyline in discrete case* where a d-dimensional object V in a data space $\mathbb{D}$ is represented by a set of multiple points in $\mathbb{D}$ called *instances* denoted by $V = \{v_1, v_2, \ldots, v_\eta\}$, where $v_i = (D_1, D_2, \ldots, D_d)$. The number of instances of an object V is denoted by $|V| = \eta$. The probability that V is a skyline is given by [37]:

$$Pr(V) = \frac{1}{\eta} \sum_{i=1}^{\eta} \prod_{W \neq V} \left(1 - \frac{|\{w \in W \mid w \prec v_i\}|}{|W|}\right) \quad (1)$$

To accommodate [37], a set of samples is drawn from the continuous range of x in D_1. These samples correspond to instances of object x. Assume that a set of five samples is collected, thus making object x has five instances where $x_1 = (1, 2)$, $x_2 = (2, 2)$, $x_3 = (3, 2)$, $x_4 = (4, 2)$, and $x_5 = (5, 2)$. On the other hand, as object y is an exact value in all dimensions, it is therefore left as it is and is assumed to have only one instance, i.e. $y_1 = (3, 1)$. We can see that instances x_3, x_4, and x_5 are dominated by y_1 (assuming that lower values are preferred). Thus, following Equation (1), we obtain the probability that x is not dominated by y as:

$$Pr(x) = \frac{1}{5} \times ((1-0) + (1-0) + (1-1) + (1-1) + (1-1)) = 0.75$$

As with the nature of continuous ranges, a random variable x can take on an uncountable infinite value in the continuous range $\left(x.D_1{}^{-} \leq x \leq x.D_1{}^{+}\right)$. Therefore, the more samples are collected in the hope of approximating the probability density function of a continuous range, the more accurate is the calculated probability. However, it is impractical and expensive (as demonstrated in the experiments conducted) to sample a colossal number of instances.

Contrariwise , Khalefa *et al.* [21] proposed *probabilistic skyline in continuous case* where a d-dimensional object v in a data space $\mathbb{D}$ with dimension μ presented as continuous range denoted by $v = ([D_l - D_u]_\mu, D_2, \ldots, D_d)$, where D_l and D_u represent the lower bound and upper bound , respectively , of object v's continuous range in dimension μ. The skyline probability of object v is defined as [21]:

$$Pr\,(v) = \int_{v \in \mathbb{D}} f\,(v) \prod_{w \neq v} \left(1 - \int_{w \prec v} f\,(w)\,dw\right) dv \tag{2}$$

However, the above exact probability computation is prohibitively expensive as we have to calculate it against every object w $(w \neq v)$ that has the potential to dominate v. Thus, Khalefa *et al.* [21] proposed four distinguishing cases to gradually calculate the probability of an object to not be dominated by another object in a pairwise manner.

To attempt this on the example of objects x and y, the exact value of object y in D_1 will be treated as range in order to conform to the structure defined in their work. Thus, we have $y = ([3:3], 1)$ while maintaining $x = ([1:5], 2)$. With this, the probability of x to be a skyline object can be computed following [21] as:

$$\begin{aligned} Pr\,(x) &= Pr\left\{x \in \left[x.D_l - y.D_l\right]\right\} \\ &\quad + \frac{1}{2} Pr\left\{x \in \left[y.D_l - y.D_u\right]\right\} \\ &= \frac{2}{4} + \frac{1}{2} \times 0 \\ &= 0.5 \end{aligned}$$

Notice that in the above equation, $Pr\{x \in [y.D_l - y.D_u]\} = 0$. This is because a continuous variable has infinite precision, thus the probability for x to assume any value in the range $(y.D_l \leq x \leq y.D_u)$, where $y.D_l = y.D_u$ is:

$$\begin{aligned} Pr\left(y.D_l \leq x \leq y.D_u\right) &= \int_{y.D_l}^{y.D_u} f\,(x)\,dx \\ &= 0 \end{aligned}$$

Although we manage to compute the probability of x in the above, notice that we are calculating the probability of x to fall within the range of $\left(x.D_l \leq x \leq y.D_l\right)$, that is $(1 \leq x \leq 3)$. This means that we are calculating as well the probability $x = (3,\ 2)$ and since y is an exact value $y = (3,\ 1)$, we know that y will always dominate x when $x = (3,\ 2)$ (as have been proven in the previous example of computing probabilistic skyline in discrete case).

Then, Li *et al.* [28] proposed the idea that a data space $\mathbb{D}$ is consisted of *interval uncertain data* when the value of its d-dimensional numeric space can be depicted as an interval that conforms to a probability distribution, or an exact value. That is, for object v with the structure of $(D_1, D_2, \ldots, D_d)$, where each D_i $(1 \leq i \leq d)$ is represented as $[v.D_i{}^{-}, v.D_i{}^{+}]$ if it is an interval number or it is represented as $[v.D_i, v.D_i]$ if it is an exact value. They hold onto the notion that, for two objects $v = \left[v.D_i{}^{-}, v.D_i{}^{+}\right]$ and $w = [w.D_i, w.D_i]$ in dimension D_i, if the center point of v is smaller than the center point of w, then $Pr\,(v \prec w) > \frac{1}{2}$. Therefore, for the aforementioned objects x and y where their median values are obtained by $x = (\frac{5+1}{2}, \frac{2+2}{2})$ and $y = (\frac{3+3}{2}, \frac{1+1}{2})$, which corresponds to $x = (3, 2)$ and $y = (3, 1)$, if all the center points of x is not bigger than y in all dimensions, and at least exist one dimension that is smaller than y, then x dominates y. From here, it is a straightforward dominance testing between two objects where it is obvious y dominates x and therefore, y is always a skyline object. Hence, they have defined skyline on *interval uncertain data* $\mathbb{D}$ as object v if there does not exist any object that dominates v. To be precise, $\{v \mid v \in \mathbb{D} \wedge (\nexists\,(w \neq v \in \mathbb{D}) \wedge w \prec v)\}$. This case in point supports as well our previous argument on the reason of computing the probability of an object v to be a skyline object by explicitly excluding any possible values where v may be dominated.

Conversely, the drawback of computing skyline on median values is that it is always assumed that all objects are represented by their median values. In reality, object with continuous ranges usually represents the uncertainty in the data collected, for instance, continuous ranges in *hotel rate per night* dimension reflect the fluctuation of the price which is dependent on the 'peak pricing'. There may be a scenario when during off-peak season, the *hotel rate per night* of hotel v might be at its lowest value, which in turn will make it a preferable choice compared to other hotels w. Or when during peak season, the *hotel rate per night* of hotel v fluctuates to its highest value possible, thus making v a bad choice as there are better hotels w with cheaper pricing. Thus, it is more practical to compute objects' probabilities rather than their expected median value in order to replicate real-world scenarios.

IV. SKYLINE PROBABILITIES OF OBJECTS IN AN *UNCERTAIN DIMENSION*

As a general rule, the uncertainty in data is expressed in term of probabilities in such a way where:

1) an object is presented with some probability. For example, a data space $\mathbb{D}$ consists of a finite set of states called *possible worlds* $\mathbb{W}$. Thus, the probability of a possible world $Pr(\mathbb{W})$ is $0 < Pr(\mathbb{W}) \leq 1$ and that $\sum_{\mathbb{W} \in \mathbb{D}} Pr(\mathbb{W}) = 1$ [45]. Or,

2) the value of a dimension is given by a probability distribution. For example, the exact value of an object in a continuous range $\mathbb{R}$ is uncertain and is modeled as a probability density function f in $\mathbb{R}$. Therefore, for any value r in $\mathbb{R}$, $0 \leq f(r) \leq 1$ and that $\int_{r \in \mathbb{R}} f(r) = 1$ [48].

We consider the second model. This would mean that we cannot simply derive precise skyline objects from these kind of datasets. Instead, we can compute the probability of each object in the dataset to be a skyline object.

Let $[v_{lb} : v_{ub}]$ denotes the continuous range of V such that $v_{lb} \leq v \leq v_{ub}$ and $\{v_{lb}, v_{ub} \in \mathbb{R} | 0 \leq v_{lb} \leq v_{ub}\}$. The probability that V will be in the continuous range $[v_{lb} : v_{ub}]$ can be defined as:

$$Pr(V \in [v_{lb} : v_{ub}]) = \int_{v_{lb}}^{v_{ub}} f(v)dv = 1 \tag{3}$$

Therefore, all probability statements about V can be answered in terms of $f(v)$, or in other words the probability density function (pdf).

Thus far, we have discussed the probability distribution of a single object with continuous range. However, the main interest of this paper is in skyline query processing which involves the probability distributions concerning two or more objects with continuous ranges. Given two objects $V \in [v_{lb} : v_{ub}]$ and $W \in [w_{lb} : w_{ub}]$, with $f(v)$ and $f(w)$ as the objects' corresponding pdfs, respectively. The probability that object V *dominates* object W can be expressed as:

$$Pr(V < W) = \int_{W \in [w_{lb} : w_{ub}]} f(w) \int_{V \prec W} f(v)\, dv\, dw \tag{4}$$

As a result, the probability of an object with continuous range to be a skyline object is the probability of that object to not be dominated by any other objects. Thus, performing skyline query on objects with continuous range is bound to produce a skyline result that is probabilistic. Logically, this would mean that any object that has a dominating probability of more than zero will have a potential to be a skyline object. In other words, the size of skyline result $\mathbb{S}$ on dataset $\mathbb{D}$ would be $\mathbb{S} \subseteq \mathbb{D}$. Hence, we employ a probability threshold τ in our *probability dominance test* in order to manage the quality and the size of skyline objects reported.

Given a set of objects $\mathbb{O}$ with continuous ranges, where it is assumed the smaller the value, the better it is, and a probability threshold τ. The probability of an object $V = [v_{lb} : v_{ub}] \in \mathbb{O}$ to be a skyline object over another object $W = [w_{lb} : w_{ub}] \in \mathbb{O}$ is the probability of V to not be dominated by W, $1 - Pr(W < V)$, and that the probability is at least τ.

Lemma 1: For an object V that is uniformly distributed over continuous range value $[v_{lb} : v_{ub}]$, $v_{lb} < v_{ub}$, it can be said with certainty that V strictly dominates W, where $[w_{lb} : w_{ub}]$, $w_{lb} < w_{ub}$, if the worst case scenario of V dominates the best case scenario of W, and V dominates W with probability that is at least τ if $[v_{lb} : v_{ub}]$ and $[w_{lb} : w_{ub}]$ intersect.

Proof: Let $v_g = v_{lb}$ represents the best case scenario of v, $v_b = v_{ub}$ represents the worst case scenario of v, and $v_a = \frac{v_{lb} + v_{ub}}{2}$ represents the average case scenario of v. Accordingly, the same is applied on object w. For $(0 \leq \tau \leq 1)$ and $\tau = 0.5$:

1) If $v_b \leq w_g$, then $Pr(v < w) = 1 > \tau$.
2) If $v_a = w_a$, then $Pr(v < w) = Pr(w < v) = \frac{1}{2} = \tau$.
3) If $v_a < w_g$, then $Pr(v < w) > \frac{1}{2} > \tau$.

From all the proofs above, we can conclude that the theory holds on. □

To extend the concept of dominance testing used in traditional skyline query to objects with continuous range is not quite that straightforward as now each dominance testing will involve probability computations, which in turn can incur an exorbitant processing cost. This is proven to be true especially when these objects are represented as multi-dimensional objects with more than one dimensions where their values are expressed as continuous range.

To ease the process of dominance testing on objects with continuous range and to avoid unnecessary probability computations, we consider the following scenarios with respect to a d-dimensional dataset $\mathbb{D} = \{V, W\}$, where V_m denotes the value of object V in the mth dimension:

1) *a pair of certain objects* V and W, where $\forall m \in \{1, 2, \ldots, d\}$, V_m and W_m are expressed as deterministic/exact values.
2) *a pair of uncertain objects* V and W, where $\exists n \in \{1, 2, \ldots, d\}$, $V_n = [v_{lb} : v_{ub}]$ and $W_n = [w_{lb} : w_{ub}]$ are expressed as continuous ranges.
3) *a pair of mismatch objects* V and W, where $\exists n \in \{1, 2, \ldots, d\}$, $V_n = [v_{lb} : v_{ub}]$ is expressed as continuous ranges, while $\forall m \in \{1, 2, \ldots, d\}$, W_m is expressed as deterministic/exact values.

Determining dominating object in scenario 1 is pretty straightforward using the traditional dominance test. Conversely, to accommodate the computation of an object's dominating probability in scenarios 2 and 3, we further define the following possible relations between two objects:

Definition 1 (Relations Between a Pair of Uncertain Objects):

Disjoint: If $w_{lb} \geq v_{ub}$

$$Pr(V < W) = 1 \tag{5}$$

Disjoint-inverse: If $w_{ub} \leq v_{lb}$

$$Pr(V < W) = 0 \tag{6}$$

Overlap: If $v_{lb} \leq w_{lb} \leq v_{ub} \leq w_{ub}$

$$Pr(V < W) = 1 - \frac{1}{2} \int_{w_{lb}}^{v_{ub}} f(v)dv \int_{w_{lb}}^{v_{ub}} f(w)dw \tag{7}$$

Overlap-inverse: If $w_{lb} \leq v_{lb} \leq w_{ub} \leq v_{ub}$

$$Pr(V < W) = \frac{1}{2} \int_{v_{lb}}^{w_{ub}} f(v)dv \int_{v_{lb}}^{w_{ub}} f(w)dw \tag{8}$$

Contain: If $v_{lb} \leq w_{lb} \leq w_{ub} \leq v_{ub}$

$$Pr(V < W) = \int_{v_{lb}}^{w_{lb}} f(v)dv + \frac{1}{2} \int_{w_{lb}}^{w_{ub}} f(v)dv \tag{9}$$

Contain-inverse: If $w_{lb} \leq v_{lb} \leq v_{ub} \leq w_{ub}$

$$Pr(V < W) = \int_{v_{ub}}^{w_{ub}} f(w)dw + \frac{1}{2}\int_{v_{lb}}^{v_{ub}} f(w)dw \quad (10)$$

Equals: If $v_{lb} = w_{lb}$ *and* $v_{ub} = w_{ub}$

$$Pr(V < W) = \frac{1}{2}\int_{v_{lb}}^{v_{ub}} f(v)dv \int_{w_{lb}}^{w_{ub}} f(w)dw \quad (11)$$

Definition 2 (Relations Between a Pair of Mismatch Objects):

Disjoint: If $w \geq v_{ub}$

$$Pr(V < W) = 1 \quad (12)$$
$$Pr(V < W) = 0 \quad (13)$$

Disjoint-inverse: If $w \leq v_{lb}$

$$Pr(V < W) = 0 \quad (14)$$
$$Pr(V < W) = 1 \quad (15)$$

Contain: If $v_{lb} \leq w \leq v_{ub}$

$$Pr(V < W) = \int_{v_{lb}}^{w-\varepsilon} f(v)\, dv \quad (16)$$
$$Pr(V < W) = \int_{w+\varepsilon}^{v_{ub}} f(v)\, dv \quad (17)$$

The two main reasons of employing $\pm\varepsilon$ as appeared in equations (16) and (17) above are (i) when using a continuous distribution to approximate an exact value, accurate approximations are likely to be acquired, and (ii) the approximation of the probability of an exact value can be obtained from the continuous distribution. Although it does not look like much the significant of $\pm\varepsilon$ in the probability calculations, the slight change in the probability of an object would affect the skyline result since we are employing the threshold τ in the *probability dominance test*.

Lemma 2: For an object V that is uniformly distributed over continuous range value $[v_{lb} : v_{ub}]$, $v_{lb} < v_{ub}$ *and an object with exact value w, it can be said with certainty that V strictly dominates W if the worst case scenario of V dominates W, and V dominates W with probability that is at least* τ *if* $[v_{lb} : v_{ub}]$ *and w intersect.*

Proof: Let $v_g = v_{lb}$ represents the best case scenario of V, $v_b = v_{ub}$ represents the worst case scenario of V, and $v_a = \frac{v_{lb}+v_{ub}}{2}$ represents the average case scenario of V. For $(0 \leq \tau \leq 1)$ and $\tau = 0.5$:

1) If $v_b < w$, then $Pr\,(V < W) = 1 > \tau$.
2) If $v_b = w$, it means we desire to compute the probability that V is strictly better than W, $Pr\,(V < W)$, which from Definition 2 we may compute this probability by $\lim_{\varepsilon\to 0} Pr\,(v \leq w - \varepsilon)$. Since $\lim_{\varepsilon\to 0}$ means that we are taking the limit as ε decreases to 0, then $Pr\,(v < w) \to 1 > \tau$.
3) If $v_a < w$, following item 3 of the proof in Lemma 1, $Pr\,(v < w) > \frac{1}{2}$ and following item 2 of the proof in this lemma, $Pr\,(v < w) \neq Pr\,(v \leq w)$ since $Pr(v \leq w)$ also includes the probability that v equals w. Therefore, $Pr\,(v < w) = \lim_{\varepsilon\to 0} Pr(v \leq w - \varepsilon) > \frac{1}{2} > \tau$.

From all the proofs above, we can conclude that the theory holds on. □

When the dominance tests are performed on objects in scenarios 2 and 3, we test a pair of objects on both of the objects' dimensions that do not contain any continuous ranges. For example, given a pair of mismatch d-dimensional objects V and W, where $W = \{w_m \in \mathbb{R}\}$ and $V = \{[v_{lb} : v_{ub}]_m | v_{lb}, v_{ub} \in \mathbb{R}, 0 \leq v_{lb} \leq v_{ub}\}$ for $m = \{1, 2, \ldots, d\}$. Then we have $\mathbb{K} = \{\forall m|$ both V and M have exact values in $m\}$ and $\mathbb{C} = \{\forall m|V$ has continuous range value and W has exact value in $m\}$. To perform dominance test between objects V and W, we begin by examining V and W in $\mathbb{K}$. If it is found in $\mathbb{K}$ for all k, $v_k \leq w_k$ and there exists k, $v_k < w_k$, then V certainly dominates W in $\mathbb{K}$, and it can be concluded that V is a candidate of skyline object regardless if V will be dominated by W in $\mathbb{C}$. This is because if V is dominated by W in $\mathbb{C}$, then it would mean that both objects are incomparable, which follows the traditional dominance test. However, if it is found in $\mathbb{K}$ there exists k, $v_k \leq w_k$ and there exists k, $w_k \leq v_k$, then it would mean that both V and W do not dominate each other (incomparable). Thus, the process of dominance testing can be terminated and both objects will be considered as skyline objects.

Hence, the process of dominance testing can be reduced to only determining the probability of W to not be dominated by V in $\mathbb{C}$ as this will be the only chance for W to be a candidate of skyline object. On the other hand, if V and W are equal in $\mathbb{K}$, then the process of dominance testing in $\mathbb{C}$ is performed in order to determine the probability of both objects to not be dominated by another. The dominance relations between V and W in $\mathbb{C}$ can be determined according to the relations defined in Definition 1 and Definition 2. As with the dominance testing in $\mathbb{K}$, if it is found that there exists object V that dominates object W in $\mathbb{C}$ in such a way that $(1 - Pr(v_c < w_c)) \geq \tau$, then it can be concluded that W has some probability of at least τ to be a candidate of skyline object.

Theorem 1: Given a threshold value τ *and a set of d-dimensional objects* $\mathbb{O} = \{V, W\}$, $W = \{w_m \in \mathbb{R}\}$ *and* $V = \{[v_{lb} : v_{ub}]_m | v_{lb}, v_{ub} \in \mathbb{R}, 0 \leq v_{lb} \leq v_{ub}\}$ *for* $m = \{1, 2, \ldots, d\}$. *The objects retrieved by the skyline query on objects* $\mathbb{O}$ *are objects that are not dominated by any other objects in* $\mathbb{K} = \{\forall m|$ *both V and M have exact values in m*$\}$ *and* $\mathbb{C} = \{\forall m|V$ *has continuous range value and W has exact value in m*$\}$, *and thus they are skyline objects with respect to* τ.

Proof: For two objects $\{V, W\} \in \mathbb{O}$, the proof consists of three parts:

1) When V dominates W in $\mathbb{K}$. If $\forall k, v_k \leq w_k$ and $\exists k, v_k < w_k$, then V definitely dominates W in $\mathbb{K}$, and thus is a candidate of skyline object regardless if the probability of V to not be dominated by W in $\mathbb{C}$ is less than τ. This is because if V dominates W in $\mathbb{K}$

but is dominated by W in $\mathbb{C}$, then both V and W are incomparable, which follows the inequality in item 2 of this proof.

2) When V and W are incomparable. If $\exists k, v_k \leq w_k$ and $\exists k, w_k \leq v_k$, then V does not dominate W, and vice-versa in $\mathbb{K}$, thus the dominance testing in $\mathbb{C}$ can be skipped.
3) When V is dominated by or equals to W in $\mathbb{K}$. If $\forall k, v_k \geq w_k$, then it is either that V is completely dominated by W or V and W do not dominate each other in $\mathbb{K}$ as they are equal. Therefore, to determine if V would still have any chance to be a candidate of skyline object, then V has to have a probability to not be dominated by W that is at least τ in $\mathbb{C}$. If $v_{[lb]c} = w_{[lb]c}$ and $v_{[ub]c} = w_{[ub]c}$, then $Pr(v_c < w_c) = Pr(w_c < v_c) = \frac{1}{2}$ and we can conclude that V is no better than W, and vice versa in $\mathbb{C}$. This means that for V to not be dominated by W in $\mathbb{C}$, then $Pr(v_c < w_c) \geq Pr(w_c < v_c)$ (since $Pr(v_c < w_c) = 1 - Pr(w_c < v_c)$), and thus $\exists c$ such that $Pr(v_c < w_c) \geq \tau$ in order for V to be considered as a candidate of skyline object.

From all the proofs above, we can conclude that if V is not dominated by other objects W in $\mathbb{K}$ or V has the probability to not be dominated by W that is more than τ in $\mathbb{C}$, then overall V is not dominated by W. Therefore, the retrieved object V is not dominated in $\mathbb{K}$ or $\mathbb{C}$, and thus is a skyline object of $\mathbb{O}$. $\square$

V. THE *SKYQUD* ALGORITHM

To demonstrate the whole process of the dominance tests on dataset containing continuous ranges we use the following example of dataset illustrated in Table 1. The dataset $\mathbb{D}$ contains a total of 40 objects that have been extracted from *Rent.com* (www.rent.com) with four main features, namely: *rent per month*, *square feet*, *number of bedrooms*, and *number of bathrooms*. The features *rent per month* and *square feet* are considered uncertain as they contained continuous ranges in their dimensions, and are represented by D_1 and D_2, respectively. On the other hand, the features *number of bedrooms* and *number of bathrooms* are considered as certain and are represented by D_3 and D_4, respectively. Formally, these dimensions are defined as $\mathbb{K} = \{D_3, D_4\}$ and $\mathbb{C} = \{D_1, D_2\}$.

We begin the process of dominance tests by partitioning the dataset according to the characteristic of each object before skyline dominance tests are performed. The partitioning process will produce at least two distinct groups, one of which will contain objects that were described in scenario 1, while the other will contain objects that were described in scenario 2. Next, these groups would either have to go through the *exact dominance test* (for objects in scenario 1) or *probability dominance test* (for objects in scenario 2). Both tests will produce a set of skyline candidates, all of which will then have to go through the *mismatch probability dominance test* that will finally produce a set of skyline objects. The reasoning behind partitioning the dataset into distinct groups is to avoid unnecessary probability computations (which would be prohibitively expensive when the number of dimensions in $\mathbb{C}$ is increasingly large).

To partition the dataset into distinctive groups, each object is examined to determine the existence of continuous ranges. Each object will have a corresponding list (denoted Θ) that will keep track of the dimensions with continuous range that exist in a particular object. Therefore, for instance, for objects $R = (r_1, [r_lb : r_ub]_2, r_3, r_4)$, $V = (v_1, [v_lb : v_ub]_2, v_3, [v_lb : v_ub]_4)$ and $W = ([w_lb : w_ub]_1, w_2, w_3, [w_lb : w_ub]_4)$, the corresponding list of dimensions with continuous range for R, V, and W would be annotated as $\Theta_r = \{2\}$, $\Theta_v = \{2, 4\}$, and $\Theta_w = \{1, 4\}$, respectively. On the other hand, for objects that are presented as an exact value in all dimensions, that is for instance, object $S = (s_1, s_2, s_3, s_4)$, since there does not exist any dimensions with continuous range in S, the corresponding list of dimensions with continuous range for S will be annotated as $\Theta_s = \{0\}$. Hence, once all objects in the dataset have been examined and the corresponding lists have been obtained, the objects will then be grouped together according to the list Θ. Following this,

TABLE 1. Running example of a list of house rentals.

ID	D_1	D_2	D_3	D_4
o_1	2008:2128	763	1	1
o_2	2013:2173	765	1	1
o_3	1415	865	1	1
o_4	1316:1908	534:639	1	1
o_5	1376:1883	586:786	1	1
o_6	1369	1003	2	1
o_7	1392:1542	1070	2	2
o_8	1429:1439	1038	2	1
o_9	1543:2808	661:690	1	1
o_{10}	1556:1616	661:690	1	1
o_{11}	1620	817	1	1
o_{12}	1660	644	1	1
o_{13}	1845	907	1	1
o_{14}	1850:2200	937	1	1
o_{15}	1443:3275	680:769	1	1
o_{16}	1850	856	1	1
o_{17}	1930:2380	865	1	1
o_{18}	2005:2265	946	1	1
o_{19}	2135:2395	901	1	1
o_{20}	2245:2395	1047	1	1
o_{21}	2260:2410	1146	1	1
o_{22}	2480	968	1	1
o_{23}	1407	1234:1363	2	2
o_{24}	1457	876:878	2	1
o_{25}	1200	444:451	1	1
o_{26}	2525	1123	1	1
o_{27}	2565	1380	1	1
o_{28}	2585:2685	1213	1	1
o_{29}	2635:2735	1192	1	1
o_{30}	2780	1182	1	1
o_{31}	2830	1153	1	1
o_{32}	1136:1196	692	1	1
o_{33}	1153	818	1	1
o_{34}	1508:3196	523:574	1	1
o_{35}	2383:3310	864:10575	2	2
o_{36}	1400	726	1	1
o_{37}	1505:2907	690	1	1
o_{38}	1325	848	1	2
o_{39}	1625	1309	1	2
o_{40}	920:980	409:455	1	1

TABLE 2. Objects in distinctive groups after partitioning.

ID	D_1	D_2	D_3	D_4
o_3	1415	865	1	1
o_6	1369	1003	2	1
o_{11}	1620	817	1	1
o_{12}	1660	644	1	1
o_{13}	1845	907	1	1
o_{16}	1850	856	1	1
o_{22}	2480	968	1	1
o_{26}	2525	1123	1	1
o_{27}	2565	1380	1	1
o_{30}	2780	1182	1	1
o_{31}	2830	1153	1	1
o_{33}	1153	818	1	1
o_{36}	1400	726	1	1
o_{38}	1325	848	1	2
o_{39}	1625	1309	1	2

(a) Group 1

ID	D_1	D_2	D_3	D_4
o_1	2008:2128	763	1	1
o_2	2013:2173	765	1	1
o_7	1392:1542	1070	2	2
o_8	1429:1439	1038	2	1
o_{14}	1850:2200	937	1	1
o_{17}	1930:2380	865	1	1
o_{18}	2005:2265	946	1	1
o_{19}	2135:2395	901	1	1
o_{20}	2245:2395	1047	1	1
o_{21}	2260:2410	1146	1	1
o_{28}	2585:2685	1213	1	1
o_{29}	2635:2735	1192	1	1
o_{32}	1136:1196	692	1	1
o_{37}	1505:2907	690	1	1

(b) Group 2

ID	D_1	D_2	D_3	D_4
o_{23}	1407	1234:1363	2	2
o_{24}	1457	876:878	2	1
o_{25}	1200	444:451	1	1

(c) Group 3

ID	D_1	D_2	D_3	D_4
o_4	1316:1908	534:639	1	1
o_5	1376:1883	586:786	1	1
o_9	1543:2808	661:690	1	1
o_{10}	1556:1616	661:690	1	1
o_{15}	1443:3275	680:769	1	1
o_{34}	1508:3196	523:574	1	1
o_{35}	2383:3310	864:10575	2	2
o_{40}	920:980	409:455	1	1

(d) Group 4

by partitioning all objects in Table 1 would yield four distinctive groups of objects as illustrated in Table 2. All objects that have been grouped together in Group 1 have the same list as $\Theta_{G1} = \{0\}$, while objects that have been grouped together in Group 2 have the same list as $\Theta_{G2} = \{1\}$. Likewise for objects in Group 3 and Group 4 where they have the same list as $\Theta_{G3} = \{2\}$ and $\Theta_{G4} = \{1, 2\}$, respectively.

By partitioning the dataset into distinctive groups, unnecessary probability computations can be avoided. For instance, based on the previous objects R, S, V, and W, the total number of dimensions with continuous range is 3, that is D_1, D_2, and D_4. Without partitioning the objects, when performing dominance testing on (S, V), for instance, we need to compute the probabilities on dimensions D_2 and D_4, while dominance testing on (R, W) would need probability computations on dimensions D_1, D_2, and D_4. Whereas by partitioning the objects, if there exists another object that is in the same group as R, then we only need to compute the probability on dimension D_2, or if there exists another object that is in the same group as S, no probability computations are needed at all. Although later on in the *mismatch probability dominance test* we still need to perform dominance testing on (S, V) and (R, W), the number of objects that has the same uncertain dimensions as R, S, V, and W has been reduced, therefore achieving the aim of the partitioning to avoid unnecessary and expensive probability computations, and this in turn would speed up the performance of *SkyQUD* algorithm.

Having different groups of objects with different representations, different dominance relation techniques and skyline probability computations are needed to cater each group as discussed below.

A. EXACT DOMINANCE TEST

If a group consists of objects that are presented as an exact value in all dimensions, then the conventional dominance testing is sufficient enough to be implemented on this group since it is such a straightforward test without having to take into account the problem of continuous ranges. The transitive relation, where $A < B$ and $B < C$, then $A < C$, summed up perfectly the process of filtering out dominated objects without having to compare with every other objects that are candidates of skyline objects. Thus, this would reduce unnecessary probability computations in future.

Based on the set of groups presented in Table 2, only Group 1 qualifies for the *exact dominance test* since the corresponding list for Group 1, $\Theta_{G1} = \{0\}$, which indicates that the presence of continuous ranges in Group 1 is non-existent. Following conventional dominance testing, we would find that o_{12} dominates o_{13}, o_{16}, o_{22}, o_{26}, o_{27}, o_{30}, and o_{31}, o_{33} dominates o_3, o_6, o_{38}, and o_{39}, while o_{36} dominates o_{11}. On the other hand o_{12}, o_{33}, and o_{36} do not dominate each other as they are incomparable since $\forall i \in (2 \leq i \leq 4)$, $o_{12}.D_i \leq o_{36}.D_i \leq o_{33}.D_i$ and $o_{33}.D_1 < o_{36}.D_1 < o_{12}.D_1$. Thus, the dominated objects will be pruned out and only o_{12}, o_{33}, and o_{36} will be the skyline candidates of Group 1 as demonstrated in Table 3. The shaded objects represent the skyline candidates of the group where these objects are not being dominated by any other objects in the group. Algorithm 1 presents the algorithm of the *exact dominance test*.

B. PROBABILITY DOMINANCE TEST

If a group consists of objects that are presented as a continuous range in at least one of the dimensions, it cannot be said with definite that an object totally dominates any other objects based solely on the traditional dominance relation theory. Therefore, the domination concept as defined in Lemma 1

TABLE 3. Skyline candidate of group 1 after *exact dominance test*.

ID	D_1	D_2	D_3	D_4
o_3	1415	865	1	1
o_6	1369	1003	2	1
o_{11}	1620	817	1	1
o_{12}	1660	644	1	1
o_{13}	1845	907	1	1
o_{16}	1850	856	1	1
o_{22}	2480	968	1	1
o_{26}	2525	1123	1	1
o_{27}	2565	1380	1	1
o_{30}	2780	1182	1	1
o_{31}	2830	1153	1	1
o_{33}	1153	818	1	1
o_{36}	1400	726	1	1
o_{38}	1325	848	1	2
o_{39}	1625	1309	1	2

Algorithm 1 Exact Dominance Test

```
Input: a set of objects in $G$
Output: a set of skyline candidates, $SkyC$
1  Initialise $SkyC$;
2  for each object $g \in G$ do
3      for each object $sc \in SkyC$ do
4          if $g \prec sc$ then
5              remove $sc$ from $SkyC$;
6          end
7          else if $sc \prec g$ then
8              remove $g$ from $G$;
9              continue step 2;
10         end
11     end
12     add $g$ to $SkyC$;
13 end
14 return $SkyC$;
```

is better suited for this group to obtain the probability of each object to be a skyline. The use of threshold value τ has been utilised in order to help prune out objects that have a skyline probability less than the threshold value.

To compute the skyline probability of each object V, the probability of object V to not be dominated by another object W, $1 - Pr(w < v)$, is computed according to the relations defined in Definition 1, where this probability has to be more than the threshold value according to Lemma 1 in order to be considered as a candidate of skyline objects. At this stage, all objects with a probability to be a skyline candidate that is lower than the threshold value will be filtered. This concept is easily applicable on objects with a single dimension with continuous range. However, if there exists more than one dimensions with continuous range, the concept as discussed in Theorem 1 is employed to determine if an object dominates another objects and if an object is a skyline.

Again, based on the set of groups presented in Table 2, Group 2, Group 3, and Group 4 qualify for the *probability dominance test* since it is found that the corresponding lists of these groups are $\forall i \in \{2, 3, 4\}, \Theta_{Gi} \neq \{0\}$. To test the dominance relations on objects with continuous ranges following Theorem 1, the dominance relation is performed first on a set of dimensions with exact values, denoted $\mathbb{K}$. The aim is to avoid unnecessary probability computations since if it is found that between two objects they are incomparable in $\mathbb{K}$, then it would mean that both objects do not dominate each other and thus can be considered as candidates of skyline objects. If it is found that an object dominates another object in $\mathbb{K}$, then the computations of probability are performed on a set of dimensions with continuous ranges, denoted $\mathbb{C}$, only on the dominated object. This concept has been deliberated in Theorem 1.

To acquire the local skyline candidates from each of these groups, the *probability dominance test* is performed on the groups separately. Let us have a look at Group 2. To start, since $\mathbb{K} = \{D_2, D_3, D_4\}$ and $\mathbb{C} = \{D_1\}$ for Group 2, for each object, the traditional dominance test is performed on $\mathbb{K}$ first. Hence for instance, between objects o_1 and o_2 as $\forall i \in \{3, 4\}$, $o_1.D_i = o_2.D_i$ and $o_1.D_2 < o_2.D_2$, then $o_{1.\mathbb{K}} \prec o_{2.\mathbb{K}}$ and we can conclude that o_1 is a possible candidate of skyline objects. Therefore, for o_2 to be a candidate of skyline objects as well, o_2 has to have the probability to not be dominated by o_1 that is at least τ in $\mathbb{C}$. This probability can be obtained by $Pr(o_{2.\mathbb{C}} < o_{1.\mathbb{C}})$ as defined by Definition 1. As we can see, the continuous ranges of o_1 and o_2 intersect in D_1 in such a way that $o_{1[lb]} < o_{2[lb]} < o_{1[ub]} < o_{2[ub]}$. The relations between o_1 and o_2 in D_1 can be seen as *overlap-inverse*. The probability of o_2 to not be dominated by o_1 in D_1 is equals to the probability of o_2 to dominate o_1, that is $(1 - Pr(o_1.D_1 < o_2.D_1)) = Pr(o_2.D_1 < o_1.D_1)$. Therefore following Eq. 8, we can obtain $Pr(o_2.D_1 < o_1.D_1)$ as follows:

$$\begin{aligned} Pr(o_2.D_1 < o_1.D_1) &= \frac{1}{2}\int_{o_{2[lb]}}^{o_{1[ub]}} f(o_2)do_2 \int_{o_{2[lb]}}^{o_{1[ub]}} f(o_1)do_1 \\ &= \frac{1}{2}\int_{2013}^{2128} f(o_2)do_2 \int_{2013}^{2128} f(o_1)do_1 \\ &= 0.34 \end{aligned}$$

Should the probability threshold τ is set to 0.5, then the *probability dominance test* would prune out object o_2 from further processing.

Following this, we can see that o_1 dominates o_2, o_7, o_8, o_{14}, o_{17}, o_{18}, o_{19}, o_{20}, o_{21}, o_{28}, and o_{29} in $\mathbb{K}$ and these objects have a probability to not be dominated by o_1 of 0.34, 1.0, 1.0, 0.62, 0.31, 0.24, 0.0, 0.0, 0.0, 0.0, and 0.0, respectively in D_1. Assuming that the probability threshold τ is set to 0.5, then the *probability dominance test* would prune these objects (except for o_7, o_8, o_{14}) from Group 2. However, o_{32} dominates o_1, o_7, o_8, and o_{14} in $\mathbb{K}$ and $\mathbb{C}$, thus removing these dominated objects from further computations as well and making o_{32} as a possible skyline candidate. Between o_{32} and o_{37} although $o_{37.\mathbb{K}} \prec o_{32.\mathbb{K}}$, yet $o_{32.\mathbb{C}} \prec o_{37.\mathbb{C}}$, which would mean both objects are incomparable and thus making both objects as skyline candidates of Group 2. Accordingly, the same concept of the *probability dominance test* is applied on Group 3 and Group 4. The shaded objects in Table 4

TABLE 4. Skyline candidates of each group after *probability dominance test.*

ID	D_1	D_2	D_3	D_4
o_1	2008:2128	763	1	1
o_2	2013:2173	765	1	1
o_7	1392:1542	1070	2	2
o_8	1429:1439	1038	2	1
o_{14}	1850:2200	937	1	1
o_{17}	1930:2380	865	1	1
o_{18}	2005:2265	946	1	1
o_{19}	2135:2395	901	1	1
o_{20}	2245:2395	1047	1	1
o_{21}	2260:2410	1146	1	1
o_{28}	2585:2685	1213	1	1
o_{29}	2635:2735	1192	1	1
o_{32}	1136:1196	692	1	1
o_{37}	1505:2907	690	1	1

(a) Group 2

ID	D_1	D_2	D_3	D_4
o_{23}	1407	1234:1363	2	2
o_{24}	1457	876:878	2	1
o_{25}	1200	444:451	1	1

(b) Group 3

ID	D_1	D_2	D_3	D_4
o_4	1316:1908	534:639	1	1
o_5	1376:1883	586:786	1	1
o_9	1543:2808	661:690	1	1
o_{10}	1556:1616	661:690	1	1
o_{15}	1443:3275	680:769	1	1
o_{34}	1508:3196	523:574	1	1
o_{35}	2383:3310	864:10575	2	2
o_{40}	920:980	409:455	1	1

(c) Group 4

represent the skyline candidates of groups 2, 3, and 4 where these objects are not being dominated by any other objects in their respective group. Algorithm 2 exemplifies the algorithm of the *probability dominance test*.

Algorithm 2 Probability Dominance Test

```
Input: a set of objects in $G$, threshold $\tau$, list $\Theta_G$
Output: a set of skyline candidates, $SkyC$
1  $\mathbb{K} = \{k | 1 \leq i \leq d, k \notin \Theta_G\}$;
2  $\mathbb{C} = \{c | c \in \Theta_G\}$;
3  Initialise $SkyC$;
4  for each object $g \in G$ do
5      for each object $sc \in SkyC$ do
6          if $g \prec sc$ in $\mathbb{K}$ then
               /* Def. 1                                  */
7              if $Pr\ (sc \prec g)$ in $\mathbb{C} < \tau$ then
8                  remove $sc$ from $SkyC$;
9              end
10         end
11         else if $sc \prec g$ in $\mathbb{K}$ then
               /* Def. 1                                  */
12             if $Pr\ (g \prec sc)$ in $\mathbb{C} < \tau$ then
13                 remove $g$ from $G$;
14                 continue step 4;
15             end
16         end
17         else if $g = sc$ in $\mathbb{K}$ then
               /* Def. 1                                  */
18             if $Pr\ (g \prec sc)$ in $\mathbb{C} < \tau$ then
19                 remove $g$ from $G$;
20                 continue step 4;
21             end
               /* Def. 1                                  */
22             else if $Pr\ (sc \prec g)$ in $\mathbb{C} < \tau$ then
23                 remove $sc$ from $SkyC$;
24             end
25         end
26     end
27     add $g$ to $SkyC$;
28 end
29 return $SkyC$;
```

C. MISMATCH PROBABILITY DOMINANCE TEST

All the objects that survived the filtering process of their own group in the *exact dominance test* and *probability dominance test* are now considered as the candidates of skyline objects. These objects however have to go through another filtering process, where they now will be compared to different groups in order to be finally accepted as skyline objects.

Mismatch probability dominance test is computed in a pairwise fashion between two local skyline candidates from different groups. This test employs probability calculations according to the relations defined in Definition 1 and Definition 2. From Theorem 1, if the computed probability for an object is below the threshold value τ, then it is no longer needed to consider the object in any further computations. In doing the pruning process, it helps to reduce the unnecessary probability computations. And thus, all objects that manage to survive the filtering process of the *mismatch probability dominance test* are considered as skyline objects.

In order to keep the comparisons between groups of local skyline candidates simple, the *mismatch probability dominance test* will treat the group G_m that has the corresponding list of uncertain dimensions $\Theta_{Gm} = \{0\}$ as a set of initial global skyline candidates, and the group will be compared to the remaining groups $G_n, n \neq m$. Therefore, for the purpose of discussing the *mismatch probability dominance test* based on the set of groups presented in Table 5, the set of global skyline candidates currently contains objects from the local skyline candidates of Group 1, which are o_{12}, o_{33}, and o_{36}. Hence, every object in groups G_n will be compared to these initial global skyline candidates.

For instance, we are performing the *mismatch probability dominance test* on local skyline candidates from Group 2 and the global skyline candidates. Between the initial skyline candidate o_{12} and object o_{32}, since the lists $\Theta_{o_{12}} = \{0\}$ and $\Theta_{G2} = \{1\}$, therefore $\mathbb{K} = \{D_2, D_3, D_4\}$ and $\mathbb{C} = \{D_1\}$. As implemented in the *probability dominance test*, when

TABLE 5. **Skyline candidates from *exact dominance test* and *probability dominance test*.**

ID	D_1	D_2	D_3	D_4
o_{12}	1660	644	1	1
o_{33}	1153	818	1	1
o_{36}	1400	726	1	1

(a) Group 1 (Global skyline candidates)

ID	D_1	D_2	D_3	D_4
o_{32}	1136:1196	692	1	1
o_{37}	1505:2907	690	1	1

(b) Group 2

ID	D_1	D_2	D_3	D_4
o_{25}	1200	444:451	1	1

(c) Group 3

ID	D_1	D_2	D_3	D_4
o_{40}	920:980	409:455	1	1

(d) Group 4

there exists dimensions with continuous range, the dominance testing is performed on $\mathbb{K}$ first, the same concept is applied in the *mismatch probability dominance test* as well. As we can see, $\forall i \in \{3, 4\}$, $o_{12}.D_i = o_{32}.D_i$ and $o_{12}.D_2 < o_{32}.D_2$, therefore we can conclude that $o_{12} \prec o_{32}$ in $\mathbb{K}$ and o_{12} still is a global skyline candidate. For o_{32} to be acknowledged as a global skyline candidate as well, o_{32} has to have the probability to not be dominated by o_{12} in $\mathbb{C}$. And since $o_{32[ub]} < o_{12}$ in D_1, then we can conclude that overall, o_{12} and o_{32} are incomparable. Similarly for objects o_{33} and o_{32} where they are both incomparable. This is because although $o_{32.\mathbb{K}} \prec o_{33.\mathbb{K}}$, o_{33} manages to not be dominated by o_{32} with a probability that is at least τ in $\mathbb{C}$. The relations between o_{32} and o_{33} in D_1 can be seen as *contain* as reflected in Definition 2. As have been elucidated in Lemma 2, when calculating the probability of a continuous range to dominate an exact value, a correction value (denoted ε) is needed to give values as small enough as possible around the exact value. Therefore, the probability of o_{33} to dominate o_{32} in D_1 can be obtained by following Eq 17 when $\varepsilon = 0.5$ as:

$$\begin{aligned} Pr(o_{33}.D_1 < o_{32}.D_1) &= \int_{o_{33}+\varepsilon}^{o_{32[ub]}} f(o_{32})do_{32} \\ &= \int_{1153+0.5}^{1196} f(o_{32})do_{32} \\ &= 0.71 \end{aligned}$$

This proves that o_{33} is not worse than o_{32} in $\mathbb{C}$ and o_{33} still is considered a global skyline candidate. Between objects o_{32} and o_{36} however, o_{32} definitely dominates o_{36} in both $\mathbb{K}$ and $\mathbb{C}$ and thus o_{36} is eliminated from the set of global skyline candidates and acknowledging o_{32} as a global skyline candidate.

Once all comparisons between the initial global skyline candidates and the local skyline candidates in Group 2 are performed, we have acquired a new set of global skyline candidates which currently contains objects o_{12}, o_{33}, and o_{32}. By implementing the *mismatch probability dominance test* in this manner, we manage to avoid redundant comparisons between objects from the same group, and the number of comparisons between objects from different groups is reduced as well. The remaining local skyline candidates in Group 3 and Group 4 will be compared to the set of global skyline candidates by applying the same concept of the *mismatch probability dominance test* accordingly. As a result, this makes o_{40} the only global skyline candidate that remain and thus the *mismatch probability dominance test* will return o_{40} as the final skyline object as shown in Table 6. Algorithm 3 demonstrates the steps involved in the *mismatch probability dominance test* in order to derive the final skyline objects. The general outline of the *SkyQUD* algorithm is presented in Algorithm 4.

TABLE 6. **Global skyline candidates after *mismatch probability dominance test*.**

ID	D_1	D_2	D_3	D_4
o_{12}	1660	644	1	1
o_{33}	1153	818	1	1
o_{36}	1400	726	1	1
o_{32}	1136:1196	692	1	1
o_{25}	1200	444:451	1	1
o_{40}	920:980	409:455	1	1

Suppose there is an algorithm that implements the same skyline probability computation as in *SkyQUD* but without having to partition the dataset into distinct groups. Then, the complexity of the algorithm is of the order $\mathcal{O}(nm)$, where n is the total number of objects in the dataset and m is the total number of skyline candidates. Therefore, it can be assumed that the complexity of *SkyQUD* is of the order $\mathcal{O}(n_z m_z)$, such that $(1 \leq z \leq \varrho)$, where $n_z < n$ is the total number of objects in group z and $m_z < m$ is the total number of skyline candidates in group z.

As the *SkyQUD* algorithm performs data partitioning on its dataset prior to any skyline computations, thus, in a worst-case scenario, the maximum number of distinctive groups created by the algorithm will be $\varrho = 2^{|\mathbb{C}|}$. Given that for an n total number of objects in a dataset with a uniform distribution of continuous data in each $d - 1$ dimension, the maximum number of objects partitioned into each distinctive group will be $G_\varrho = \frac{n}{2^{|\mathbb{C}|}}$. Following this, given that in a worst-case scenario, none of the objects dominated each other in each group, then the maximum number of skyline candidates for each distinctive group will be $SkyC_\varrho = \frac{n}{2^{|\mathbb{C}|}}$. On the other hand, in the *mismatch probability dominance test*, the maximum number of skyline candidates at the initial stage will be $SkyC = n - \frac{n}{2^{|\mathbb{C}|}}$, as it combines all the skyline candidates obtained from each distinctive group (except for the skyline candidates obtained from the *exact dominance test*), while the maximum number of global skyline candidates will be $GSkyC = \frac{n}{2^{|\mathbb{C}|}}$. At the end of the *mismatch probability dominance test* stage, both the *SkyC* and *GSkyC* will be n, respectively, in the worst-case scenario.

Algorithm 3 Mismatch Probability Dominance Test

```
Input: a set of local skyline candidates
       $SkyC = \{SkyC_1, SkyC_2, \ldots, SkyC_\varrho\}$, threshold $\tau$
Output: a set of objects, $Sky$, which is the skyline with
        probability $\geq \tau$
1  Initialise $GSkyC$;
2  add $SkyC_m$, where $\Theta_{Gm} = \{0\}$, to $GSkyC$;
3  for each group $SkyC_n \in SkyC$, $(1 \leq n \leq \varrho)$, $n \neq m$ do
4      for each object $s \in SkyC_n$ do
5          for each object $sc \in GSkyC$ do
6              $\mathbb{K} = \{k | 1 \leq k \leq d, k \notin \Theta_{Gn}, k \notin \Theta_{sc}\}$;
7              $\mathbb{C} = \{c | 1 \leq c \leq d, c \in \Theta_{Gn}, c \in \Theta_{sc}\}$;
8              if $s \prec sc$ in $\mathbb{K}$ then
                   /* Def. 1, 2 */
9                  if $Pr(sc \prec s)$ in $\mathbb{C} < \tau$ then
10                     remove $sc$ from $GSkyC$;
11                 end
12             end
13             else if $sc \prec s$ in $\mathbb{K}$ then
                   /* Def. 1, 2 */
14                 if $Pr(s \prec sc)$ in $\mathbb{C} < \tau$ then
15                     remove $s$ from $SkyC_n$;
16                     continue step 4;
17                 end
18             end
19             else if $s = sc$ in $\mathbb{K}$ then
                   /* Def. 1, 2 */
20                 if $Pr\ (s \prec sc)$ in $\mathbb{C} < \tau$ then
21                     remove $s$ from $SkyC_n$;
22                     continue step 4;
23                 end
                   /* Def. 1, 2 */
24                 else if $Pr\ (sc \prec s)$ in $\mathbb{C} < \tau$ then
25                     remove $sc$ from $GSkyC$;
26                 end
27             end
28         end
29     end
30     add $SkyC_n$ to $GSkyC$;
31 end
32 add $GSkyC$ to $Sky$;
33 return $Sky$;
```

VI. EMPIRICAL STUDY

In this section, the results of the extensive experiment to examine the algorithms proposed in this paper are reported. All experiments were conducted on a PC with Intel Core i5-3470 3.20GHz processor and 7.8GB main memory running Ubuntu Linux operating system.

To fairly evaluate the performance of *SkyQUD* when processing skyline queries on *uncertain dimensions*, we conduct extensive experiments on *SkyQUD* and compare *SkyQUD* to *USky* [21], *BBIS* [28], *BU* [37], and *TD* [37].

Note that for *BU* and *TD*, we have sampled a number of random points for each object with continuous ranges to

Algorithm 4 *SkyQUD*

```
Input: a $d$-dimensional dataset $S$ with
       $\mathbb{K} = \{D_i\}$, $1 \leq i \leq d$, threshold $\tau$
Output: a set of objects, $Sky$, which is the skyline with
        probability $\geq \tau$
1  Initialise $Sky$, $SkyC$, and $G$;
2  $G$ = apply object partitioning;
3  for each group $G_i \in G$, $(1 \leq i \leq \varrho)$ do
4      if $\Theta_{Gi} = \{0\}$ then
5          $SkyC$ = apply Algorithm 1;
6      end
7      else if $\Theta_{Gi} \neq \{0\}$ then
8          $SkyC$ = apply Algorithm 2;
9      end
10 end
11 $Sky$ = apply Algorithm 3 on $SkyC$;
12 return $Sky$;
```

represent as instances of the object in order to simulate the datasets used within these two algorithms. We use the same parameters described in [37] to generate the instances. The instances of each object with continuous range are generated according to a uniform distribution within the continuous range of the object and the number of instances of an object with continuous range is distributed uniformly within the range $[1, u]$, where u is 400 by default. Therefore, in general, on average each object with continuous range has $\frac{u}{2}$ instances, and thus the total number of instances in a dataset is $\frac{nu}{2}$, which is 200,000,000 by default (given that n is 1,000,000 by default). However, objects with exact value are left as they are without any manipulation and are assumed to have only one instance. Thus, the total number of instances generated for objects with continuous range is $\frac{n\delta u}{2}$, which is 100,000,000 by default (given that the distribution of objects with continuous range or exact value, denoted δ, is 0.5 by default). Nevertheless, note that we did not compare with *BU* and *TD* in terms of number of pairwise comparisons as in our work, it means performing dominance tests between objects, while in [37] the dominance tests would instead be performed between instances of objects and this would be unfair on the *BU* and *TD* algorithms.

The same goes for *BBIS* where the algorithm performed comparisons between entries in the R*-tree, and these entries may be an intermediate entry or a leaf entry. The leaf entries correspond to the original objects in the dataset, while the intermediate entries correspond to the minimum bounding rectangles (MBRs) of nodes at the lower level. Therefore, it would be unfair to measure *BBIS* in terms of number of pairwise comparisons.

On the other hand, since all objects in *USky* are assumed to be presented as continuous ranges in the same dimension, that is, the datasets used in *USky* have the structure of $([a_{lb} : a_{ub}]_1, a_2, \ldots, a_d)$, therefore, *USky* will treat every object with exact value in *uncertain dimensions* as continuous ranges by assigning their lower and upper bounds

TABLE 7. Summary of experiment parameters for *SkyQUD*.

Parameter	Dataset type	
	Synthetic	NBA
n	**1M**, 2M, 4M, 6M, 8M, 10M	2k, 4k, 6k, 8k, 10k, 12k, 14k, 16k, **21,961**
τ	0.1, 0.2, 0.3, 0.4, **0.5**, 0.6, 0.7, 0.8, 0.9, 1.0	
δ (%)	10, 20, 30, 40, **50**, 60, 70, 80, 90	
d	3, 6, **10**, 15, 20	6, 10, 15, **17**

to be that of the objects' exact values. In words, suppose an object $v = (5, 3, \ldots, 7)$, *USky* will treat the object as $v = ([5 : 5], 3, \ldots, 7)$.

Following [21], we generate a d-dimensional synthetic dataset of n objects with the following settings. We set d to be varied from 3 to 20 and n to be varied from 1M to 10M. Each dimension is represented with a uniform random variable from 1 to 10,000. We have set the first dimension to be the dimension that will represent the concept of *uncertain dimension* (i.e. $\mathbb{C}(D_1)$), where the distributions (δ) between exact values and continuous ranges in $\mathbb{C}(D_1)$) by default is set to be equally distributed. On the other hand, the NBA statistics contains records of 21,961 NBA players from the year 1946 to 2009. Each record has 16 dimensions that represent various statistics associated with basketball games such as game played, total points, total rebound, total assist, point per game, etc. However, the NBA statistics are initially represented in exact values that are certain and complete, and therefore, following [21], we have to explicitly add another dimension to introduce the concept of *uncertain dimension* to the dataset. The *uncertain dimension* is generated following the same settings used by [21] in generating synthetic datasets.

Table 7 briefly summarises the experiment parameters used in this section for both the synthetic and NBA datasets, and unless mentioned otherwise, we set the default value of the threshold value τ to 0.5, the distributions of exact values $\delta_{\mathbb{K}}$ and continuous ranges $\delta_{\mathbb{C}}$ in a dimension to 50%, and the number of tested dimensions d to 10 and 17 for the synthetic and real datasets, respectively. We also set the default size of the synthetic and real datasets n to 1M and 21,961, respectively. These default values are shown in bold font.

A. SCALABILITY

Fig. 5 and Fig. 6 show the scalability of our algorithm in terms of number of pairwise comparisons and CPU time, respectively, when increasing the data size n (objects) from 2k to 20k for real dataset and from 1M to 10M for synthetic datasets. However, *BU* and *TD* fail to terminate on synthetic datasets with $n >$ 1M (Fig. 6c, Fig. 6d, and Fig. 6e) due to the massive number of instances generated $\left(\frac{n\delta u}{2}\right)$. The performance of both *BU* and *TD* is clearly worse than the other algorithms on NBA dataset (Fig. 6a) due to the tremendous amount of instances $\left(\frac{n\delta u}{2}\right)$ to be processed as compared to the number of objects (n) processed in the other algorithms. Therefore, we evaluate the *SkyQUD* algorithm in comparison to the *USky* and *BBIS* algorithms in terms of their CPU time (Fig. 6) and the number of pairwise comparisons performed (Fig. 5) in order to process skyline query on dataset with *uncertain dimensions*. *SkyQUD* shows a better performance as during pairwise comparison between objects, the cost of dominance testing is different for each group of objects (see Sect. V-A and Sect. V-B).

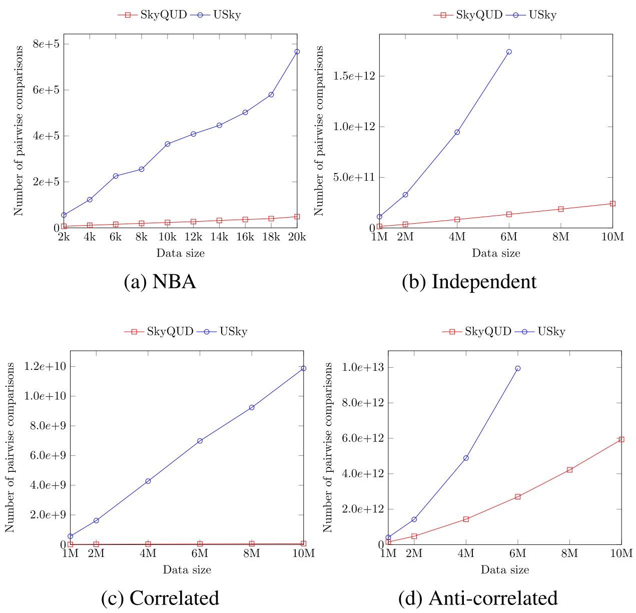
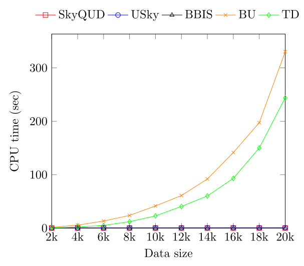


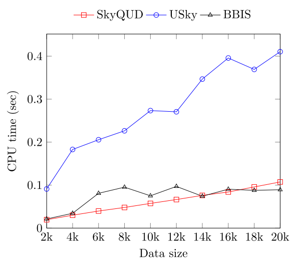
FIGURE 5. Effect of *n* on number of pairwise comparisons.

In *SkyQUD*, the cost has been significantly reduced as the dominance testing that is performed in the *exact dominance test* is applied only on $n\delta$ objects and does not require any probability computation. This is identical to the situations of processing skylines on certain objects. Only $n\delta$ objects do require skyline probability computations. The experiments clearly indicate that the pruning technique in *SkyQUD* has significantly saved the cost of computing skyline probabilities on large data size. Meanwhile in the *USky*, *BU*, and *TD* algorithms, the dominance tests require the computation of objects' probabilities on n and $\left(\frac{n\delta u}{2}\right)$ objects, respectively, instead of only $n\delta$. The number of comparisons in *SkyQUD* is smaller than all the other algorithms due to the implementation of the partitioning method as the more objects that have been eliminated in each group, the little it is the dominance tests that have to be performed on the skyline candidates. Generally, anti-correlated dataset has the largest number of pairwise comparisons and the longest CPU time while correlated dataset has the smallest number of pairwise comparisons and the fastest CPU time. This too is similar to the situations of processing skylines on certain objects.

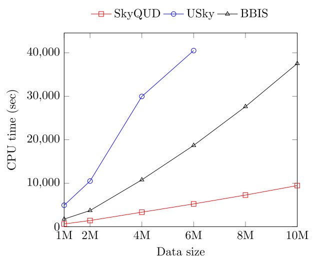
On the other hand, the *SkyQUD* algorithm managed to outperform the *BBIS* algorithm on all synthetic datasets (Fig. 6) due to the implementation of an R*-tree index structure in the *BBIS* algorithm to index the datasets. Although R*-tree is well known for its I/O optimal performance (and this has undoubtedly contributed to the fast performance of *BBIS* as the algorithm only needs to perform a single access to those nodes that may contain skyline objects [28]), however,



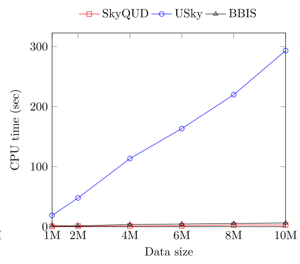
(a) NBA



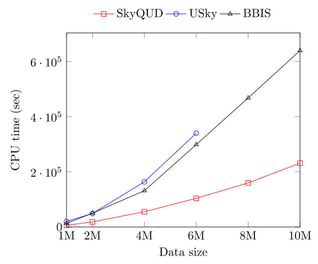
(b) NBA (zoomed)



(c) Independent

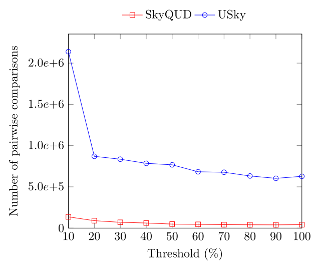


(d) Correlated

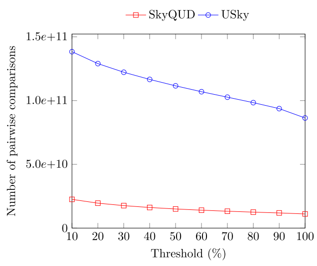


(e) Anti-correlated

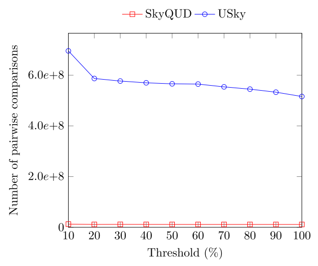
FIGURE 6. Effect of *n* on CPU time.



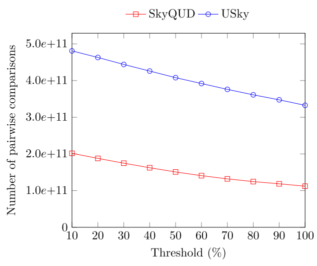
(a) NBA



(b) Independent



(c) Correlated



(d) Anti-correlated

FIGURE 7. Effect of τ on number of pairwise comparisons.

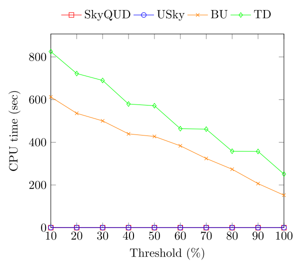
the performance of R*-tree is known to deteriorate as the number of dimensions in the dataset increases higher than five [4], [28], [35]. Nevertheless, the performance of R*-tree is not necessarily dependent solely on the number of dimensions, but the size of dataset plays an important role as well. As demonstrated in Fig. 6b, *BBIS* managed to outperform *SkyQUD* even when the number of dimensions for the NBA dataset is set to 17. This is due to the size of its dataset that is quite small (i.e. 21,961), which indicates that the dataset is sparser than the synthetic datasets. In sparse dataset, the bounding boxes of each node in the R*-tree have smaller chance to overlap, thus making it easier to prune groups of dominated objects based on their bounding boxes.

Also, since the continuous ranges in the *uncertain dimension* in all the datasets have been uniformly but randomly generated, thus, as the data size grew, there are times when the overlapping of the data becomes unpredictable, which in turn has affected the performance of an algorithm. If there are more data that are overlapping, then the algorithm would be required to perform probability computation on the overlapping data, which consumes time. Conversely, if there are less overlapping data, then the algorithm can quickly eliminates dominated data from further computation without having to perform any probability computation. This situation has been reflected in Fig. 6b, Fig. 8d, Fig. 8e, Fig. 9a, Fig. 9c, Fig. 10d, and Fig. 10e, where the *USky* and *BBIS* algorithms' performance on the CPU time is increasing or decreasing unsteadily. The reason that the *SkyQUD* algorithm is largely unaffected by this situation is due to the *exact dominance test* that was performed prior to the *probability dominance test*, where $n\delta$ of data has been quickly compared and pruned out without having to perform any probability computation.

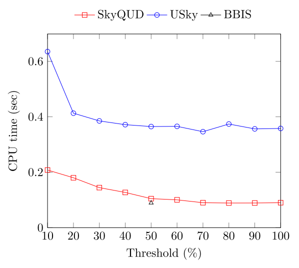
B. EFFECT OF THRESHOLD

Fig. 7 and Fig. 8 present the effect of increasing the threshold value τ in terms of number of pairwise comparisons and CPU time, respectively, when the threshold value is varied from 0.1 to 1.0 for both synthetic and real datasets. Logically, the threshold value is used to filter out all objects that have the probability to be a skyline candidate that is less than τ. Thus, as the threshold value τ increases, all algorithms exhibit an increment of speed in their performance as well. Still, *SkyQUD* displays a better performance than *USky*, *BU*, and *TD*, as other than the use of threshold to help filter objects efficiently, the implementation of the *exact dominance test* on $n\delta$ objects would ensure a faster filtering time (as discussed earlier), since skyline queries on certain objects consume lesser time than probabilistic skyline queries as the traditional skyline queries do not require any probability calculations. Therefore, the *exact dominance test* method is impervious to the change of the threshold value.

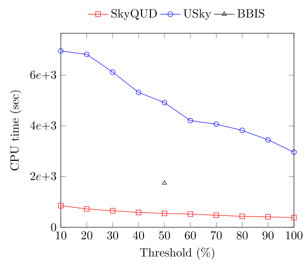
Regarding the number of pairwise comparisons, the results clearly indicate that *SkyQUD* performs fewer dominance tests than *USky* as the decision of elimination through the dominance tests on $n\delta$ of objects with exact value is guaranteed, while in *USky* the decision of elimination through the dominance tests on n objects is dependent on the threshold value. On the other hand, the threshold values do not give any impact on the performance of *BBIS* as the algorithm does not implement any probability computations. Nonetheless,



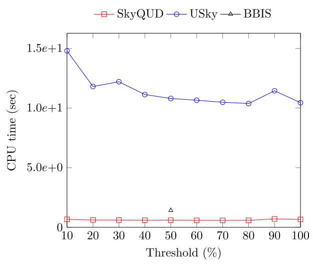
(a) NBA



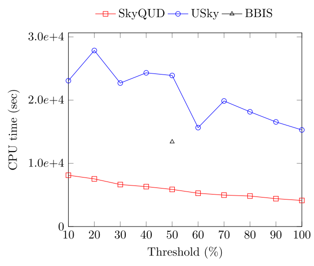
(b) NBA (zoomed)



(c) Independent

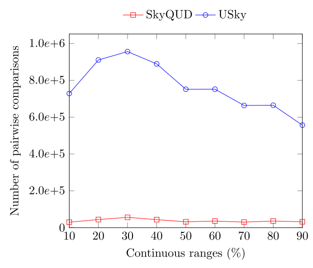


(d) Correlated

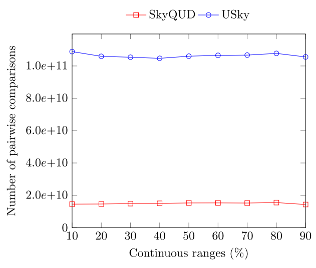


(e) Anti-correlated

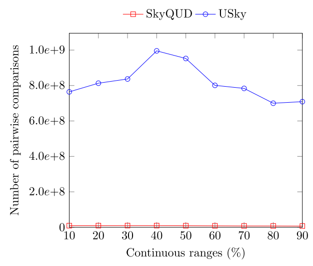
FIGURE 8. Effect of τ on CPU time.



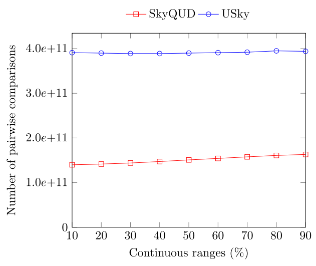
(a) NBA



(b) Independent

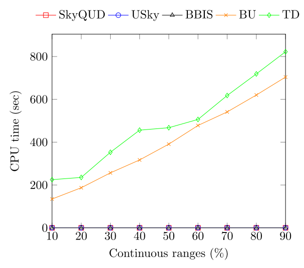


(c) Correlated

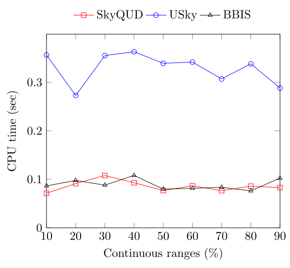


(d) Anti-correlated

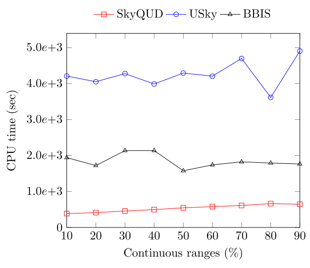
FIGURE 9. Effect of δ on number of pairwise comparisons.



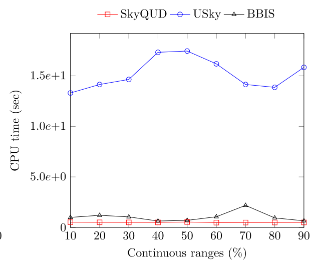
(a) NBA



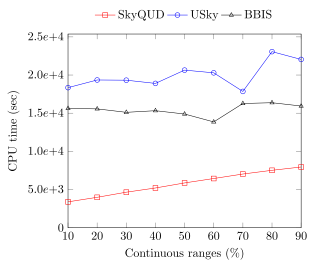
(b) NBA (zoomed)



(c) Independent



(d) Correlated



(e) Anti-correlated

FIGURE 10. Effect of δ on CPU time.

to be impartial, we still experimented on *BBIS* and *SkyQUD* to compare their performances in terms of CPU time. We set the parameter for *SkyQUD* to be 50% for the threshold value and tested it against *BBIS* (Fig. 8). The reason being that *BBIS* uses the median value for every continuous range, as this median value typically reflects the average-case scenario. To conclude, even though the threshold value does not affect the *BBIS* algorithm, yet, *SkyQUD* managed to outperform *BBIS* on all datasets.

C. EFFECT OF DATA DISTRIBUTION

The behaviour of *SkyQUD* and *USky* illustrated in Fig. 9 demonstrated their performance in terms of number of pairwise comparisons. While the behavior of *SkyQUD*, *USky*, *BBIS*, *BU*, and *TD* illustrated in Fig. 10 demonstrated the algorithms' performances against one another in terms of CPU time. Both experiments are conducted and observed when the distributions of data are varied in such a manner where the distributions of data are increased from $\delta_{\mathbb{C}} = 10\%$ and $\delta_{\mathbb{K}} = 90\%$ to $\delta_{\mathbb{C}} = 90\%$ and $\delta_{\mathbb{K}} = 10\%$. Based on the results observed, both *USky* and *BBIS*'s performances are not greatly affected by the data distributions. Yet, in general, the performance of *USky* is not really affected by the distributions of continuous ranges and exact values.

Similarly, because *BBIS* treat every continuous range as an exact value, it would mean that *BBIS* does not need to compute any object probabilities. Therefore, the existence of continuous ranges does not give any impact on

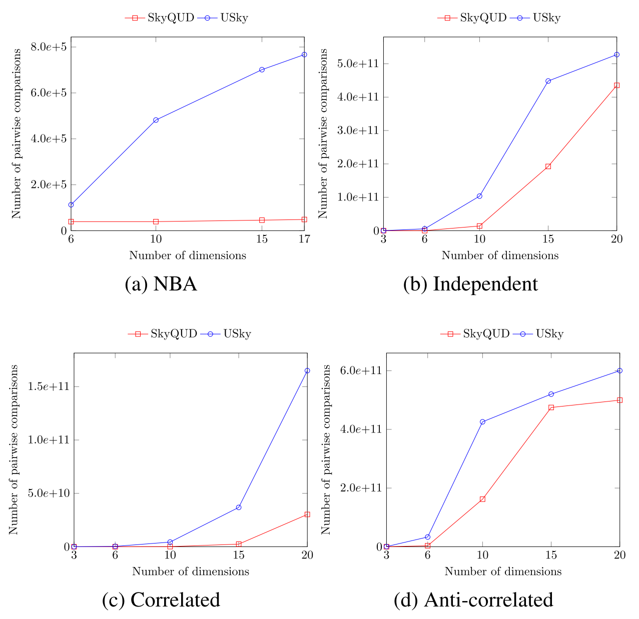


FIGURE 11. Effect of *d* on number of pairwise comparisons.

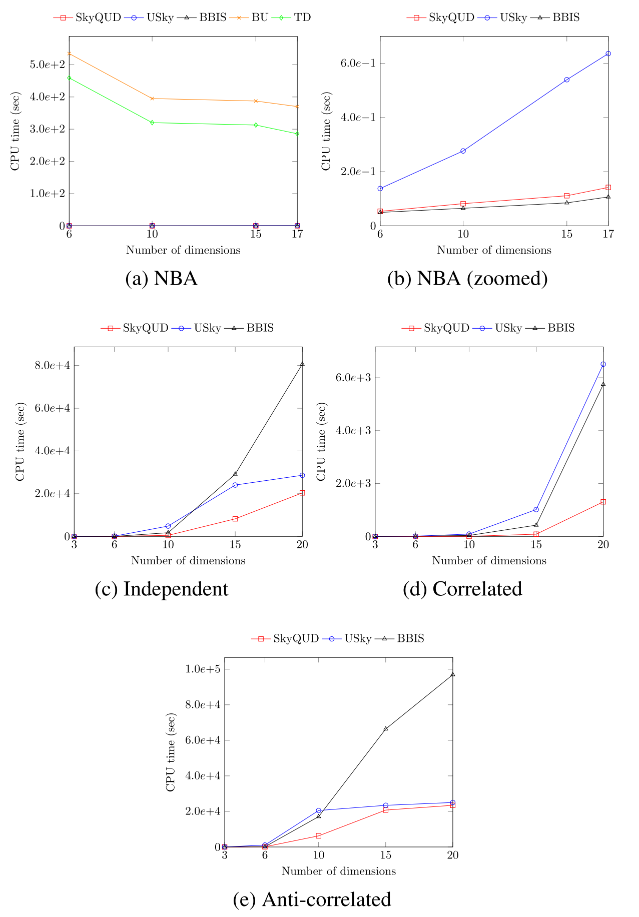


FIGURE 12. Effect of *d* on CPU time.

BBIS's performance. Even so with this advantage on the *USky* and *BBIS* sides, *SkyQUD* succeeded to outperform both algorithms. This success can be attributed to the implementation of data partitioning in *SkyQUD*, and the poor performance of R*-tree (as elucidated in Sect. III). The implementation of the *probability dominance test* has contributed as well to the fast performance of *SkyQUD*. This is because, the probability computations are executed on groups with *uncertain dimensions* and only when one object clearly dominates or equals to the other object in $\mathbb{K}$. This has aided in the speed of *SkyQUD* as unnecessary probability computations can be avoided. Contrarily, as both *BU* and *TD* perform skyline processing on instances of objects, therefore, all data with continuous range, that is $\delta = \delta_{\mathbb{C}}$, will be used to generate the corresponding instances. This would mean the higher the $\delta_{\mathbb{C}}$ generated, the higher the number of instances ($\frac{n\delta u}{2}$), which in turn reduces the speed of *BU* and *TD*, and thus, greatly affected their performance.

D. EFFECT OF DIMENSIONALITY

This experiment gives the effect of varying the number of dimensions in *SkyQUD*, *USky*, *BBIS*, *BU*, and *TD* as it is well known that skyline query is highly influenced by the number of dimensions. Fig. 11 and Fig. 12 show the number of pairwise comparisons and the CPU time of *SkyQUD*, *USky*, and *BBIS* as the number of dimensions is varied from 3 to 20 on 1M objects for synthetic datasets, and from 6 to 17 on 21,961 objects for NBA dataset. All three algorithms follow similar trends where the CPU time and the number of pairwise comparisons increase with the increase of the number of dimensions. This is due to the increase in the cost of dominance testing between objects, as when the number of dimensions increases the sparser the datasets will be, and lesser chance for objects to overlap which cause in a lot of incomparable objects.

Conversely, in terms of speed, *BBIS* outperforms *SkyQUD* when the number of dimensions is below than 6. However, as it has been proven that R*-tree could not index data with more than five dimensions adequately [4], [28], [35], it has indirectly diminished the performance of *BBIS* as the number of dimensions increased from 6 to 20. The deteriorating performance is because the higher the number of dimensions, the more regions are overlapping in the directory. This occurrence will affect the query performance as multiple paths have to be traversed even for a simple search query [4], [28], [35]. Nevertheless, it must be noted that this issue is not a specific problem that is faced by an R*-tree index structure but then is a general problem in indexing high-dimensional data [4]. On the other hand, *BU* and *TD* can perform faster when the dimensionality increases. The excellent performance is proven true as when the dimensionality increases, the dataset becomes sparser. In turn, it has decreased the averaged number of possible dominating objects for each object. Nevertheless, the performance of *BU* and *TD* still could not outperform the other algorithms in comparison. This scenario is illustrated in Fig. 12a.

VII. CONCLUSION

Data uncertainty generally occurs in web data extractions, autonomous web databases, decision-making system, and recommendation system. In these database systems, uncertainties may occur in one or more dimensions and are unavoidable, hence applying the conventional skyline technique is impractical and can lead to prohibitive processing cost and inaccurate results. Therefore, we have defined the concept of *uncertain dimensions* and proposed an efficient algorithm, *SkyQUD*, to answer skyline query on data with *uncertain dimensions*. When data is represented as continuous ranges, the result of the skyline query is bound to be probabilistic as each object is associated with a probability value of it being a query answer. Therefore, we have designed the algorithm to be able to accept a threshold value that is specified by users, in which each object must exceed. We presented the *SkyQUD* algorithm in a three-phase context, namely: *exact dominance test*, *probability dominance test*, and *mismatch probability dominance test*, which follows the filter-refine approach. These phases are preceded with a partitioning process to group the data into distinctive groups. These phases compute each object's probability for being a skyline object by avoiding unnecessary probability computations. We have conducted extensive experiments and exhibited the results to demonstrate the efficiency of the *SkyQUD* algorithm to tolerate skyline queries on data with uncertain dimensions.

REFERENCES

[1] M. J. Atallah and Y. Qi, "Computing all skyline probabilities for uncertain data," in *Proc. 28th ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst. (PODS)*, 2009, pp. 279–287.

[2] I. Bartolini, P. Ciaccia, and M. Patella, "SaLSa: Computing the skyline without scanning the whole sky," in *Proc. 15th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2006, pp. 405–414.

[3] I. Bartolini, P. Ciaccia, and M. Patella, "The skyline of a probabilistic relation," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1656–1669, Jul. 2013.

[4] S. Berchtold, D. A. Keim, and H. P. Kriegel, "The X-tree: An index structure for high-dimensional data," in *Proc. 22nd Int. Conf. Very Large Data Bases (VLDB)*, 1996, pp. 28–39.

[5] A. Bhattacharya, P. Teja, and S. Dutta, "Caching stars in the sky: A semantic caching approach to accelerate skyline queries," in *Proc. Int. Conf. Database Expert Syst. Appl. (DEXA)*, 2011, pp. 493–501.

[6] C. Böhm, F. Fiedler, A. Oswald, C. Plant, and B. Wackersreuther, "Probabilistic skyline queries," in *Proc. 18th ACM Conf. Inf. Knowl. Manage. (CIKM)*, 2009, pp. 651–660.

[7] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, 2001, pp. 421–430.

[8] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2006, pp. 503–514.

[9] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proc. 19th Int. Conf. Data Eng.*, 2003, pp. 717–816.

[10] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou, "Parallel distributed processing of constrained skyline queries by filtering," in *Proc. IEEE 24th Int. Conf. Data Eng.*, Apr. 2008, pp. 546–555.

[11] G. B. Dehaki, H. Ibrahim, F. Sidi, N. I. Udzir, and A. A. Alwan, "A rule-based skyline computation over a dynamic database," in *Proc. 22nd Int. Conf. Inf. Integr. Web-Based Appl. Services*, Nov. 2020, pp. 97–103.

[12] E. Dellis and B. Seeger, "Efficient computation of reverse skyline queries," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2007, pp. 291–302.

[13] X. Ding, X. Lian, L. Chen, and H. Jin, "Continuous monitoring of skylines over uncertain data streams," *Inf. Sci.*, vol. 184, no. 1, pp. 196–214, Feb. 2012.

[14] Z. Dzolkhifli, H. Ibrahim, F. Sidi, L. S. Affendey, S. N. M. Rum, and A. A. Alwan, "A skyline query processing approach over interval uncertain data stream with K-means clustering technique," in *Proc. 11th Int. Conf. Adv. Databases, Knowl., Data Appl. (DBKDA)*, 2019, pp. 51–56.

[15] S. Elmi, K. Benouaret, A. Hadjali, M. A. B. Tobji, and B. B. Yaghlane, "Computing skyline from evidential data," in *Proc. 8th Int. Conf. Scalable Uncertainty Manage. (SUM)*, 2014, pp. 148–161.

[16] S. Elmi, M. A. B. Tobji, A. Hadjali, and B. B. Yaghlane, "Selecting skyline stars over uncertain databases: Semantics and refining methods in the evidence theory setting," *Appl. Soft Comput.*, vol. 57, pp. 88–101, Aug. 2017.

[17] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2005, pp. 229–240.

[18] B. Groz and T. Milo, "Skyline queries with noisy comparisons," in *Proc. 34th ACM SIGMOD-SIGACT-SIGAI Symp. Princ. Database Syst.*, May 2015, pp. 185–198.

[19] X. Han, J. Li, D. Yang, and J. Wang, "Efficient skyline computation on big data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 11, pp. 2521–2535, Nov. 2013.

[20] Y.-K. Huang, "Continuous d_ε-skyline queries for objects with time-varying attribute in road networks," *Proc. IEEE 31st Int. Conf. Adv. Inf. Netw. Appl.*, May 2017, pp. 439–446.

[21] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline query processing for uncertain data," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2010, pp. 1293–1296.

[22] D. Kim, H. Im, and S. Park, "Computing exact skyline probabilities for uncertain databases," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 12, pp. 2113–2126, Dec. 2012.

[23] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2002, pp. 275–286.

[24] R. D. Kulkarni and B. F. Momin, "Parallel skyline computation for frequent queries in distributed environment," in *Proc. Int. Conf. Comput. Techn. Inf. Commun. Technol. (ICCTICT)*, Mar. 2016, pp. 374–380.

[25] T. M. N. Le, J. Cao, and Z. He, "Answering skyline queries on probabilistic data using the dominance of probabilistic skyline tuples," *Inf. Sci.*, vols. 340–341, pp. 58–85, May 2016.

[26] K. C. K. Lee, W.-C. Lee, B. Zheng, H. Li, and Y. Tian, "Z-SKY: An efficient skyline query processing framework based on Z-order," *VLDB J.*, vol. 19, no. 3, pp. 333–362, Jun. 2010.

[27] X. Li, K. Ren, X. Li, and J. Yu, "Efficient skyline computation over distributed interval data," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 10, pp. 1–19, 2017.

[28] X. Li, Y. Wang, X. Li, and G. Wang, "Skyline query processing on interval uncertain data," in *Proc. IEEE 15th Int. Symp. Object/Compon./Service-Oriented Real-Time Distrib. Comput. Workshops*, Apr. 2012, pp. 87–92.

[29] J. Liu, X. Li, K. Ren, and J. Song, "Parallelizing uncertain skyline computation against n-of-*N* data streaming model," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 4, p. e4848, Feb. 2019.

[30] M. L. Ma'aruf, H. Ibrahim, N. F. M. Sani, and R. Yaakob, "Skyline computation of uncertain database: A survey," in *Proc. 6th Int. Conf. Comput. Inform. (ICOCI)*, 2017, pp. 84–90.

[31] M. L. Ma'aruf, H. Ibrahim, N. F. M. Sani, and R. Yaakob, "An indexed non-probability skyline query processing framework for uncertain data," in *Proc. Int. Conf. Adv. Mach. Learn. Technol. Appl.*, 2020, pp. 289–301.

[32] M. M. Lawal, H. Ibrahim, N. F. M. Sani, and R. Yaakob, "Analyses of indexing techniques on uncertain data with high dimensionality," *IEEE Access*, vol. 8, pp. 74101–74117, 2020.

[33] T. De Matteis, S. Di Girolamo, and G. Mencagli, "Continuous skyline queries on multicore architectures," *Concurrency Comput., Pract. Exper.*, vol. 28, no. 12, pp. 3503–3522, Aug. 2016.

[34] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, 2003, pp. 467–478.

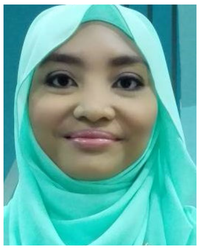
[35] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 41–82, Mar. 2005.

[36] J. Pei, W. Jin, M. Ester, and Y. Tao, "Catching the best views of skyline: A semantic approach based on decisive subspaces," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2005, pp. 253–264.

[37] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2007, pp. 15–26.

[38] N. H. M. Saad, H. Ibrahim, F. Sidi, and R. Yaakob, "Non-index based skyline analysis on high dimensional data with uncertain dimensions," in *Proc. Int. Baltic Conf. Databases Inf. Syst. (DB IS)*, 2018, pp. 272–286.

[39] N. H. M. Saad, H. Ibrahim, F. Sidi, R. Yaakob, and A. A. Alwan, "Computing range skyline query on uncertain *dimension*," in *Database Expert Syst. Applications* (Lecture Notes in Computer Science), vol. 9828, S. Hartmann and H. Ma, Eds. Cham, Switzerland: Springer, 2016, pp. 377–388.
[40] N. H. M. Saad, H. Ibrahim, F. Sidi, R. Yaakob, and A. A. Alwan, "Reporting skyline on uncertain dimension with query interval," *J. Telecommun., Electron. Comput. Eng.*, vol. 8, no. 6, pp. 17–21, 2016.
[41] N. H. M. Saad, H. Ibrahim, F. Sidi, and R. Yaakob, "A framework for evaluating skyline query over uncertain autonomous databases," in *Proc. Int. Conf. Comput. Sci. (ICCS)*, 2014, pp. 1546–1556.
[42] C. Sheng and Y. Tao, "Worst-case I/O-efficient skyline algorithms," *ACM Trans. Database Syst.*, vol. 37, no. 4, pp. 1–24, 2012.
[43] M. A. Siddique and Y. Morimoto, "K-dominant skyline computation by using sort-filtering method," in *Proc. 13th Pacific–Asia Conf. Adv. Knowl. Discovery Data Mining*, 2009, pp. 839–848.
[44] H. Z. Su, E. T. Wang, and A. L. P. Chen, "Continuous probabilistic skyline queries over uncertain data streams," in *Proc. Int. Conf. Database Expert Syst. Appl. (DEXA)*, 2010, pp. 105–121.
[45] D. Suciu, D. Olteanu, C. Ré, and C. Koch, *Probabilistic Databases*. San Rafael, CA, USA: Morgan & Claypool, 2011.
[46] K. L. Tan, P. K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2001, pp. 301–310.
[47] Y. Tao, X. Xiao, and J. Pei, "SUBSKY: Efficient computation of skylines in subspaces," in *Proc. 22nd Int. Conf. Data Eng. (ICDE)*, 2006, pp. 65–74.
[48] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha, "Updating and querying databases that track mobile units," *Distrib. Parallel Databases*, vol. 7, no. 3, pp. 257–387, 1999.
[49] P. Wu, C. Zhang, Y. Feng, B. Zhao, D. Agrawal, and A. Abbadi, "Parallelizing skyline queries for scalable distribution," in *Proc. Int. Conf. Extending Database Technol. (EDBT)*, 2006, pp. 112–130.
[50] M. L. Yiu and N. Mamoulis, "Efficient processing of top-k dominating queries on multi-dimensional data," in *Proc. 33rd Int. Conf. Very Large Data Bases (VLDB)*, 2007, pp. 483–494.
[51] H. Yong, J.-H. Kim, and S.-W. Hwang, "Skyline ranking for uncertain data with maybe confidence," in *Proc. IEEE 24th Int. Conf. Data Eng. Workshop*, Apr. 2008, pp. 572–579.
[52] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, "Efficient computation of the skyline cube," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2005, pp. 241–252.
[53] Y. Zeng, G. Chen, K. Li, Y. Zhou, X. Zhou, and K. Li, "M-Skyline: Taking sunk cost and alternative recommendation in consideration for skyline query on uncertain data," *Knowl.-Based Syst.*, vol. 163, pp. 204–213, Jan. 2019.
[54] N. Zhang, C. Li, N. Hassan, S. Rajasekaran, and G. Das, "On skyline groups," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 942–956, Apr. 2014.
[55] W. Zhang, X. Lin, Y. Zhang, W. Wang, and J. X. Yu, "Probabilistic skyline operator over sliding windows," in *Proc. Int. Conf. Data Eng. (ICDE)*, 2009, pp. 1060–1071.
[56] J. Zheng, J. Chen, and H. Wang, "Efficient geometric pruning strategies for continuous skyline queries," *ISPRS Int. J. Geo-Inf.*, vol. 6, no. 3, pp. 1–28, 2017.
[57] W. Zheng, L. Zou, X. Lian, L. Hong, and D. Zhao, "Efficient subgraph skyline search over large graphs," in *Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, Nov. 2014, pp. 1529–1538.
[58] X. Zhou, K. Li, Y. Zhou, and K. Li, "Adaptive processing for distributed skyline queries over uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 2, pp. 371–384, Feb. 2016.



NURUL HUSNA MOHD SAAD received the bachelor's degree in computer science and the Ph.D. degree in database systems from Universiti Putra Malaysia (UPM), Malaysia, in 2010 and 2018, respectively. Her current research interests include databases, knowledge-based systems, and cyber security issues.

HAMIDAH IBRAHIM (Member, IEEE) received the Ph.D. degree in computer science from the University of Wales Cardiff, U.K., in 1998. She is currently a Full Professor with the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). Her current research interests include databases (distributed, parallel, mobile, biomedical, and XML) focusing on issues related to integrity maintenance/checking, ontology/schema/data integration, ontology/schema/data mapping, cache management, access control, data security, transaction processing, query optimization, query reformulation, preference evaluation–context-aware, information extraction, concurrency control, and data management in mobile, grid, and cloud.

FATIMAH SIDI received the Ph.D. degree in management information system from Universiti Putra Malaysia (UPM), Malaysia, in 2008. She is currently working as an Associate Professor in computer science with the Department of Computer Science, Faculty of Computer Science and Information Technology, UPM. Her current research interests include knowledge and information management systems, data and knowledge engineering, and database and data warehouse.

RAZALI YAAKOB received the bachelor's and master's degrees in computer science from Universiti Putra Malaysia, in 1996 and 1999, respectively, and the Ph.D. degree from the University of Nottingham, U.K., in 2008. He is currently a Lecturer with the Faculty of Computer Science and IT, Universiti Putra Malaysia. He is a member of the Faculty of the Intelligent Computing Group. His research interests include artificial neural networks, pattern recognition, and evolutionary computation in game playing.

ALI A. ALWAN received the master's and Ph.D. degrees in computer science from Universiti Putra Malaysia (UPM), Malaysia, in 2009 and 2013, respectively. He is currently an Associate Professor with the Kulliyyah (Faculty) of Information and Communication Technology, International Islamic University Malaysia (IIUM), Malaysia. His research interests include preference queries, skyline queries, probabilistic and uncertain databases, query processing and optimization and management of incomplete data, data integration, location-based social networks (LBSN), recommendation systems, and data management in cloud computing.