Computing and Informatics, Vol. 32, 2013, 987–1012

PROBABILISTIC SKYLINE QUERIES OVER UNCERTAIN MOVING OBJECTS

Xiaofeng Ding, Hai Jin, Hui Xu

Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science
Huazhong University of Science and Technology
Wuhan, Hubei, P. R. China, 430074
e-mail: {xfding, hjin, chinahui1988}@hust.edu.cn

Wei Song

School of Internet of Things
Jiangnan University
Wuxi, Jiangsu, P. R. China, 214122
e-mail: waterbottle9988@hotmail.com

Communicated by János Fodor

Abstract. Data uncertainty inherently exists in a large number of applications due to factors such as limitations of measuring equipments, update delay, and network bandwidth. Recently, modeling and querying uncertain data have attracted considerable attention from the database community. However, how to perform advanced analysis on uncertain data remains an interesting question. In this paper, we focus on the execution of skyline computation over uncertain moving objects. We propose a novel probabilistic skyline model where an uncertain object may take a probability to be in the skyline at a certain time point, therefore a p-t-skyline contains those moving objects whose skyline probabilities are at least p at time point t. Computing probabilistic skyline over a large number of uncertain moving objects is a daunting task in practice. In order to efficiently compute the probabilistic skyline query, we propose a discrete-and-conquer strategy, which follows the sampling-bounding-pruning-refining procedure. To further reduce the skyline computation cost, we propose an enhanced framework that is based on a multi-

dimensional indexing structure combined with the discrete-and-conquer strategy. Through extensive experiments with synthetic datasets, we show that the framework can efficiently support skyline queries over uncertain moving object and is scalable on large data sets.

Keywords: Mobile computing, probabilistic skyline query, uncertain data

Mathematics Subject Classification 2010: 68-P10

1 INTRODUCTION

In many applications data values are inherently uncertain. Consider a scientific data application system where measurements taken from sensors or other devices are analyzed for particular usage. In such a system, data with known degrees of errors are often exhibited due to limitations of bandwidth and the battery power of devices. In the case of a moving object database, it is infeasible for the database to track the movement of object and store the exact locations of object at all times. Due to the importance of those applications and the rapidly increasing amount of uncertain data collected, analyzing large collections of uncertain data has become an important research topic. In particular, how to perform advanced analysis on uncertain data in moving object environments, especially the skyline analysis, remains an interesting and challenging problem [1, 2].

Since the introduction of skyline operator in the database community by Börzsönyi et al. [3], several efficient algorithms have been proposed for general skyline processing, and most of these studies have showed that skyline analysis is very important in applications such as multi-criteria decision making system. In particular, these algorithms utilize techniques such as divide-and-conquer [3], nearest neighbor search [4], sorting [5], and indexing structures [8, 9] to answer the general skyline queries. Furthermore, several studies have also focused on the skyline query processing in a variety of environmental settings such as data streams [1], data residing on mobile devices [10], uncertain data [2], spatial data [7], and moving object [6]. However, to the best of our knowledge, no work has studied the probabilistic skyline queries over uncertain moving object. The most relevant work are the continuous skyline queries methods for moving object [6] and the bottom-up algorithm proposed by Pei et al. [2], which can be utilized to address probabilistic skyline queries by considering it as a special case of dynamic skyline queries. Since [2] is addressing the static problem, it overlooks the mobile properties of moving object, thus its performance is not optimal in moving environments. On the other hand, [6] cannot deal with the data with uncertainty.

In the current moving object environments, due to the inherent uncertainty in data values, providing meaningful answer seems impossible. However, one can state with probability that object o is in the skyline at a certain time point t. That is to

say, the uncertainty of the moving object may not allow us to determine an object as a "precise" skyline. Instead, more object could have the opportunities of being in the "possible" skyline result. Consider the situations in real-time applications such as e-games and digital battle systems; due to the imprecise positions of moving players, the probabilistic skyline query is needed when one fighting player wants to keep track of those enemies who are close and most dangerous in terms of multiple aspects like energy, weapon, strategy, etc.

The conception of probabilistic answers for skyline queries over uncertain data was first introduced in [2]. Motivated by this, we answer p-t-skyline queries over uncertain moving object using a set which contains those moving object whose skyline probabilities are at least p at a user specified time t. That is to say, in addition to identifying these moving object, the probability of each object being in the skyline is also calculated and returned. The probabilities allow the users to have an appropriate confidence in the answer set instead of having an incorrect answer or no answer at all. Furthermore, it is an easy task for users who want to choose those object whose skyline probabilities are greater than a threshold at a certain time point. Note that, since the domination relationship can be determined with respective to both spatial (dynamic) and non-spatial (static) attributes of object P, we are focusing on the spatial one and the proposed method can be easily extended to the non-spatial one.

In this paper, we present a novel technique for providing probabilistic guarantees to answers of skyline queries. Our objective is to minimize the total calculation costs. As an overview, the algorithm first eliminates all those objects that have no chance of being in the skyline. Then, for every object that may be the skyline point, its probability is evaluated. As mentioned above, there is lack of any detailed study on probabilistic skyline over uncertain moving object.

The rest of this paper is organized as follows. First, we define the problem of *Probabilistic Skyline Query* (PSQ) and list the notations used throughout the paper in Section 2. Then, in Section 3 we establish the theoretical foundation of our proposed algorithm and discuss the optimal solutions for PSQ. The performance of our proposed algorithm is evaluated in Section 4. The related work to skyline queries and probabilistic skyline queries is presented in Section 5. Finally, we conclude the paper and discuss our future work in Section 6.

2 FORMAL PROBLEM DEFINITION

In the following sections, we first define the general skyline query using notations given in Table 1. Then, we introduce the probabilistic skyline query based on the definition of a general skyline query.

2.1 General Skyline Query

Given a set of d-dimensional points P, a point p dominates another p', donated by $p \prec p'$, if the coordinate of p on each dimension is not larger than that of p',

Notations	Meaning
P	set of static database points
MO	set of uncertain moving object
M, N	uncertain moving object
$\|MO\|$	cardinality of MO
pdf	probability density function
x, x'	instances (positions) of uncertain moving object
$x \prec x'$	instance x dominates x'
$Pr(x \prec x', t)$	the probability that x dominates x'

Table 1. The summary of notations

and is strictly smaller on at least one dimension. Generally, a skyline set consists of those points which are not dominated by others. For example, Figure 1 shows a scenario where each point in the 2-dimensional space (x, y) corresponds to a hotel record. The x-dimension represents the price of the hotel room, and the y-dimension captures its distance from the beach. Obviously, hotel p_1 dominates p_2 because the former is cheaper in price and closer to the beach, so that p_1 is more preferable according to any preference function that is monotone on the two axes [3]. The skyline of all the points shown in Figure 1 contains 3 points p_1, p_3, and p_5, which offers different tradeoffs between price and distance: p_5 is the nearest to the beach, p_3 is the cheapest, and p_1 may be a good compromise of the two factors. Notice that every point in the skyline does not need to dominate a point of P. For instance, the point p_1 dominates five other points and p_5 dominates two other points, but the point p_3 dominates no point.

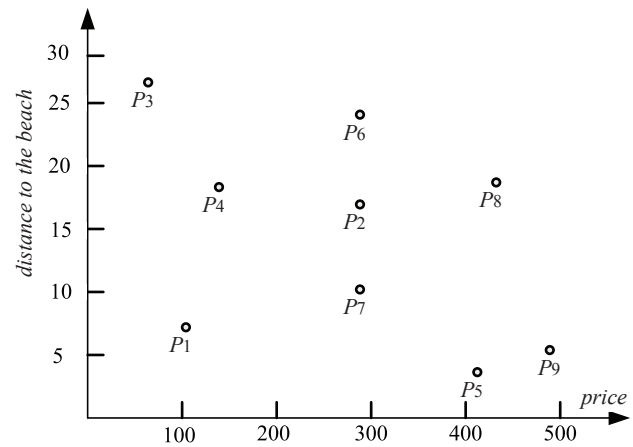


Figure 1. A general skyline example

2.2 Probabilistic Skyline Query

According to the definitions given before [15, 16, 17, 18, 22], an uncertain moving object M is conceptually associated with

1. a d-dimensional uncertain region $U(M,t)$,
2. a moving velocity $V(M,t)$, and
3. a probability density function $pdf(x,t)$,

where x is an arbitrary d-dimensional point.

Definition 1. The uncertain region of M at time t, denoted by $U(M,t)$, is a closed region where M is located at time t.

Definition 2. The moving velocity of M at time t, denoted by $V(M,t)$, is a d-dimensional vector which determines the moving direction.

Definition 3. The uncertain probability density function (pdf) of object M, denoted by $pdf(x,t)$, is a pdf of s location x at time t, that has a value of 0 outside $U(M,t)$.

Since $pdfi(x,t)$ is a probability density function, it has the property

$$\int_{x \in U(M,t)} pdf(x,t)\mathrm{d}x = 1.$$

Based on these new conditions, the data in the database is only an estimate of the actual location at most points in time – an object can be anywhere within its uncertain region (the grey area in Figure 2). Without loss of generality, for a large set of moving object, we assume each object is independent. That is, an instance of an object does not depend on the instances of any other object. Moreover, we assume that, for an uncertain moving object, each instance takes the same probability to happen, that is, the *pdf* follows a uniform distribution. However, our model can be easily extended to cases where dependencies (e.g., correlations) exist among object and another kind of distribution about *pdf* exists such as normal distribution.

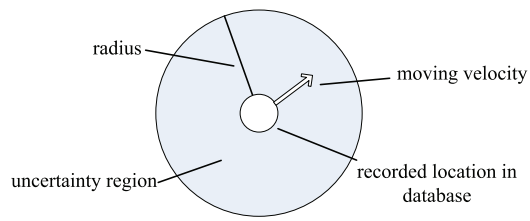


Figure 2. An uncertain moving object example

Given a set MO of uncertain moving object defined above, let M and N be two moving object contained in MO, $pdf(x,t)$ and $pdf'(x',t)$ be the corresponding uncertain probability density functions of M and N, respectively. Then, the definition of dominant relationship over two uncertain moving object can be generalized as follows:

Definition 4. The probability that M dominates N at time t is:

$$\begin{aligned} Pr(M \prec N, t) &= \int_{x' \in U(N,t)} pdf'(x',t) \sum_{\forall x \in U(M,t), x \prec x'} p(M,x) dx' \\ &= \int_{x' \in U(N,t)} pdf'(x',t) \left(\int_{SR(M,t)} pdf(x,t) \mathrm{d}x \right) \mathrm{d}x' \end{aligned} \quad (1)$$

where x and x' are the possible positions of M and N, $P(M,x)$ is the probability of M within position x, $SR(M,t)$ is the sub-region of $U(M,t)$ and for any position x within $SR(M,t)$, x dominates x' holds.

Based on the probabilistic dominance relationship defined above, it is straightforward to deduce the probability that an uncertain moving object M is in the skyline at a certain time point:

Definition 5. The probability that M is in the skyline at time t is:

$$\begin{aligned} Pr(M,t) &= \int_{m \in U(M,t)} pdf(m,t) \left(\prod_{\forall N \in MO \setminus \{M\}} (1 - Pr(N \prec m, t)) \right) \mathrm{d}m \\ &= \int_{m \in U(M,t)} pdf(m,t) \left(\prod_{\forall N \in MO \setminus \{M\}} \left(1 - \int_{SR(N,t)} pdf'(x',t) \mathrm{d}x' \right) \right) \mathrm{d}m \end{aligned} \quad (2)$$

where $SR(N,t)$ is the sub-region of $U(N,t)$ and for any position x' within $SR(N,t)$, x' dominates m holds.

Intuitively, consider an uncertain moving object M with uncertain probabilistic density function $pdf(x,t)$. The probability that M appears at position m within uncertain region $U(M,t)$ is given by $P(M,m) = pdf(m,t)$. For any other object N with probability density function $pdf'(x',t)$, the probability that N dominates m is the integral of the pdf function over region $SR(N,t)$ within where each position x' dominates m always holds, that is $Pr(N \prec m, t) = \int_{SR(N,t)} pdf'(x',t) \mathrm{d}x', \forall x' \in SR(N,t), x' \prec m$. Thus, for each possible dominating object of m, we calculate the probability that m is not dominated by multiplying the probabilities that each dominating object is not in the position with dominant power.

As a result, an uncertain moving object may take a probability to be in the skyline. Based on the above definitions it is easy to extend the notion of skyline to probabilistic skyline:

Definition 6. For a set of uncertain moving objects MO and a probability threshold $p(0 \leq p \leq 1)$, a p-t-skyline returns those objects of which the skyline probabilities are at least p at time t. That is,

$$Skyline(p,t) = \{M \in MO, Pr(M,t) \geq p\}.$$

A naive brute-force search algorithm for finding the p-t-skyline of a set MO requires examining all moving object in MO against each other. In other words, for each moving object M, its skyline probability at time t is computed according to Equation (2). The time complexity of this naive algorithm is $O(|MO|^2)$ as it exhaustively examines all the moving object against others.

Therefore, computing p-t-skyline through the naive method incurs expensive costs, especially when the dimensionality and cardinality of datasets are high. On the other hand, an optimal PSQ algorithm must examine each moving object M against only those object which are inside the possible dominating region of M. In the next section, we establish the theoretical foundation of our efficient PSQ algorithm which allows us to eliminate a majority of the non-qualifying moving object without calculating their skyline probabilities.

3 THE DISCRETE-AND-CONQUER METHOD

In this section, we describe the discrete-and-conquer strategy for probabilistic skyline queries with the objective to reduce the total computing cost. The main idea is to follow the *sampling-bounding-pruning-refining* procedure. The first three steps form an iteration to filter out these moving object in the database that have no chance of being in the p-t-skyline, and this iteration goes on until every uncertain moving object can be determined in the p-t-skyline set or not. The last step, namely the refining step, is an important part of our solution: it computes the probability of being the p-t-skyline for each moving object that remains after the first three steps.

In particular, in the execution of discrete-and-conquer framework for answering PSQ, we compute and refine the bounds of instances of uncertain moving object by selectively computing the skyline probabilities of a small subset of instances. An uncertain moving object may be pruned or validated using the skyline probabilities of its instances, or those of some other object.

3.1 Sampling Instances

For an uncertain moving object M, its instances may have the same dominating object or may vary dramatically from one instance to another. Whether M is in the p-t-skyline or not should be determined by computing the skyline probabilities of as few instances of M as possible. However, selecting an optimal subset of instances to compute is not an easy task since, without computing the exact probabilities of each instance, it is difficult to know their distribution. Motivated by this, we propose a layer-by-layer heuristic sampling method.

According to the equation in Section 2, among all instances of the uncertain moving object M, it is important to find the instances that are not dominated by any other instances, i.e., those instances listed in layer-1 as illustrated in Figure 3, and then compute their corresponding skyline probabilities.

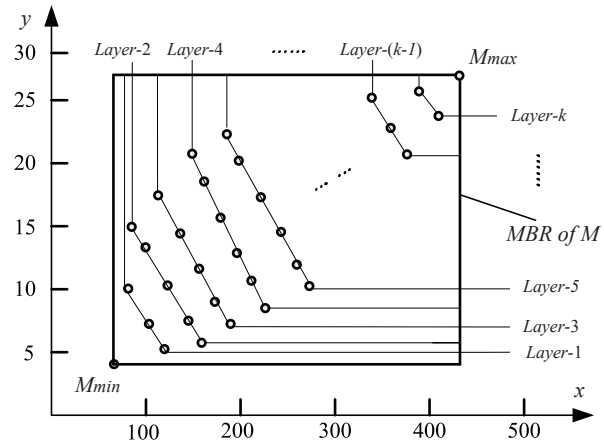


Figure 3. Layer and MBR of object M at time t

Formally, for an uncertain moving object M, an instance $m \in M$ is assigned to layer-1 if m is not dominated by any other instances in M. Accordingly, an instance v is assigned to layer-k ($k > 1$) if v is not at the $1^{\text{st}}, \ldots, (k-1)^{\text{th}}$ layers, and v is not dominated by any instances except for those at the $1^{\text{st}}, \ldots, (k-1)^{\text{th}}$ layers.

Obviously, the higher the number of layers where instances are located, the higher probability of the corresponding instances being dominated. That is to say, the skyline probabilities of the instances at layer-1 can serve as the upper bounds of the skyline probabilities of other instances, and can be used to generate an upper bound of the skyline probability of M.

Given a number n of points, which are generated according to the probability density function of M at time t. The way that these instances are assigned into layers within $U(M,t)$ will affect the efficiency of the query procedure. In the following, a quick assigning method will be introduced to speed up the query efficiency.

First, we define a key for each instance m, and the key of the instance is defined as its Euclidean distance from the original point. Then, we sort all the instances in their key ascending order. This is motivated by the SFS Algorithm [2, 5]. Based on this rationale, the sorted list of instances has a good property: for any instances x and m such that $x \prec m$, x precedes m in the final list.

After the sorted list is constructed, it is sequentially scanned to get the instances on each layer. That is to say, the first returned instance with the minimum key value is assigned to layer-1. The second returned instance is compared with the first one.

If the second one is dominated, then it is assigned to layer-2; otherwise it is assigned to layer-1.

However, as the list size increases, it is inefficient to compare the current instance with all the previously assigned instances in every layer. Certain layers should be chosen to help determine the status of the current processed instance. That is, for processing an instance m, suppose there are already h layers. We compare m with the instances assigned to layer-$\lceil h/2 \rceil$, and then one of the following two cases may happen. If m is dominated by an instance at that layer, then m must be assigned to some layer higher than $\lceil h/2 \rceil$. On the other hand, m is neither dominated by, nor dominates, any instance at that layer. Then, m must be at that layer or at some lower layer. This "binary search" like method is conducted recursively until m is assigned to a layer.

3.2 Bounding Skyline Probabilities

After all the instances of moving object M are selected and assigned, we can make a minimum bounding rectangle (MBR for short) of M, which includes all its instances at time t. In particular, let $M_{\min}$(the lower-left corner) and $M_{\max}$(the upper-right corner) be the minimum and maximum corner of the MBR of M at time t, respectively, as illustrated in Figure 3. Note that $M_{\min}$ and $M_{\max}$ may not be two actual instances of M. In this case, we treat them as virtual instances and define their skyline probabilities for special usage:

$$Pr(M_{\min}, t) = \prod_{N \neq M} \left(1 - \int_{x' \prec M_{\min}} pdf'(x', t)\mathrm{d}x'\right) \tag{3}$$

And

$$Pr(M_{\max}, t) = \prod_{N \neq M} \left(1 - \int_{x' \prec M_{\max}} pdf'(x', t)\mathrm{d}x'\right). \tag{4}$$

Based on the skyline probabilities of some particular instances, we define the following lemma:

Lemma 1 (Bounding Skyline Probabilities [2])**.** Let M be an uncertain moving object and $m_i (1 \leq i \leq n)$ be the instances of M at time t:

1. if $m_i \prec m_j (1 \leq i, j \leq n)$, then $Pr(m_i, t) \geq Pr(m_j, t)$.
2. $Pr(M_{\min}, t) \geq Pr(M, t) \geq Pr(M_{\max}, t)$.

The proof of this lemma is obvious, since the instance, which dominates the other instances, has a larger skyline probability. Furthermore, Lemma 1 provides an approach to compute the upper and lower bounds of instances or uncertain moving object by using the skyline probabilities of other instances.

According to the first inequality in the lemma, the skyline probability of an instance can be bounded by those instances dominating or dominated by it. In other

words, when the skyline probability of an instance is calculated, the bounds of the skyline probabilities of some other instances of the same object may be refined accordingly.

The second inequality indicates that the minimum and maximum corners of the MBR can determine the upper and lower bounds of the skyline probability for an uncertain moving object M, and thus can be used for pruning or validating purpose. In the next section, the pruning and validating techniques will be presented.

3.3 Pruning and Validating Techniques

Based on the dominant relationship between object or instances, if the skyline probability of an uncertain moving object or an instance is computed, we can use this information to prune the other uncertain instances or object. Generally, it is possible to use this information to prune a non-qualifying moving object, or to validate an object that indeed satisfies the probabilistic skyline query, without computing its accurate skyline probabilities. According to Lemma 1, we immediately have the following observation to determine the p-t-skyline membership of an uncertain moving object using the minimum or maximum corner of its MBR.

Observation 1. For an uncertain moving object M and a probabilistic skyline query with threshold p at time t:

1. If $Pr(M_{\min}, t) < p$, then M can be pruned for there is no chance of M being in the skyline;
2. If $Pr(M_{\max}, t) \geq p$, then M is guaranteed to satisfy the query and is in the p-t-skyline.

It is also possible to use the information about one uncertain object to prune other uncertain instances or object. First, if an instance m of an uncertain moving object M is dominated by the maximum corner of another uncertain moving object N at time t, then m cannot be in the p-t-skyline. Second, if the minimum corner of an uncertain moving object N is dominated by an instance m of other uncertain moving object M at time t and $Pr(m, t) < p$, then N cannot be in the p-t-skyline. Furthermore, we can also use the information about skyline probabilities of some instances of an uncertain moving object to prune some other uncertain moving object. The following observation will specify the detailed pruning techniques:

Observation 2. For two uncertain moving objects M, N and a probabilistic skyline query with threshold p at time t, let $m_i (1 \leq i \leq n)$ be all the instances of M, M' be a subset instances of M at time t such that M'_{max} dominates the minimum corner of MBR of N, and m be the number of instances of M that dominate M'_{max}:

1. If $\frac{m}{n} > 1 - p$, then N can be pruned for there is no chance of N being in the p-t-skyline;
2. If $\frac{m}{n} \leq 1 - p$, then N can be neither pruned nor validated for the p-t-skyline;

Proof. Formally, since $N_{\min}$ is dominated by $M'_{\max}$, each instance of N is dominated by the instances within M'. Thus, only when M appears outside of M', $N_{\min}$ is not dominated, that is to say, the probability that N is not dominated by M at most equals to $(1-m/n)$, if $(m/n) > 1-p$, then $(1-m/n) < p$, so the skyline probability of N is smaller than p, then N can be pruned. For the situation $(1-m/n) \geq p$, the other object except M may dominate N, so the dominant relationship is pending; thus the observation holds. □

In many cases, an uncertain moving object can be asserted to violate or satisfy a query using the above rules directly, thus avoiding the expensive skyline probability computation, which is necessary only if these observations can neither prune nor validate the object. The precise skyline computation method will be introduced in the following refinement step.

3.4 Refinement Strategies

Since the execution time for the probabilistic skyline query is significantly affected by the number of uncertain moving object that need to be considered. We use the above iteration to filter out a majority of the moving objects that are not in the p-t-skyline. For the remaining object that may take a chance of being in the skyline, the skyline probabilities should be calculated precisely. In this section, we study how Equation (2) presented in Section 2 can be efficiently applied to calculate the skyline probabilities of the remaining object.

According to Equation (2), for an uncertain moving object M, we have to find all other moving object that may contain some instances dominating M at query time. Those object are called possible dominating object (*pdo*) of M and are stored in a queue called $pdo(M,t)$. The skyline membership of M depends only on those possible dominating object. All other object that do not contain any instances dominating M need not be considered.

To speed up the search of possible dominating object, we organize the minimum corners of MBRs of all uncertain moving object into a global VCI-tree [11]. Then we issue a window query with the origin and M_{max} as the opposite corners on the global VCI-tree. The possible dominating object for M are selected only when the minimum corner of the object is located within the query window, and is assigned to its queue $pdo(M,t)$.

In order to calculate the skyline probability of M, we have to compute the skyline probability for each instance m of M first. We compare m with all the possible dominating object of M in $pdo(M,t)$ one by one. To facilitate the comparison and speed up the computing time, we incrementally maintain a local VCI-tree contains every object in $pdo(M,t)$. When an instance $m \in M$ is compared with object in $pdo(M,t)$, only those object with instances that have a key value less than m may dominate m. Then, we issue a window query with the origin and m as the opposite corners on the local VCI-tree to compute possible dominating object of m at time t, which are also stored in a queue called $pdo(m,t)$.

Since the probability that an object N dominates m is $\int_{x' \prec m} pdf'(x', t)\mathrm{d}x'$ and $pdf'(x', t)$ is a uniform probability density function, we can deduce that:

$$\begin{aligned} Pr(N \prec m, t) &= \int_{x' \prec m} pdf'(x', t)\mathrm{d}x' \\ &= \int_{SR(N,t)} pdf'(x', t)\mathrm{d}x' \\ &= \frac{dominating\ area\ of\ m\ SR(N,t) \bigcap U(N,t)}{uncertain\ region\ of\ N\ at\ time\ t\ (U(N,t))} \end{aligned}$$

Thus, the problem of calculating the probability that N dominates m is reduced to the problem of finding the overlapping area of dominating region of m (a rectangle composed by m and the original point) and uncertain region of N at time t. Here, we present the calculation of $Pr(N \prec m, t)$ by assuming m is located outside $U(N, t)$.

Figure 4 shows the overlapping area of dominating region of m and uncertain region of N at time t when m is outside of $U(N, t)$. We can get:

$$\cos\theta = \frac{m_y - N_y}{R_n}, \quad \text{then } \theta = \arccos\frac{m_y - N_y}{R_n}.$$

The grey area above m_y (the red line) can then be evaluated as follows:

$$\frac{1}{2}R_n^2(2\theta) - \frac{1}{2}R_n^2\sin(2\theta).$$

Since the area of $U(N, t)$ is πR_n^2, we have:

$$\begin{aligned} Pr(N \prec m, t) &= \int_{x' \prec m} pdf'(x', t)\mathrm{d}x' \\ &= \left[\pi R_n^2 - \left(\frac{1}{2}R_n^2(2\theta) - \frac{1}{2}R_n^2\sin(2\theta)\right)\right] / \pi R_n^2 \\ &= 1 - \left(\frac{1}{2\pi}(2\theta) - \frac{1}{2\pi}\sin(2\theta)\right). \end{aligned}$$

3.5 Implementation Framework

In order to speed up the procedure of answering PSQ, we are going to use an index structure for storing all the uncertain moving object. As our problem definitions assume that the uncertain moving object follows the linear movement, which is the most popular movement model in the literature. This model assumes that the moving object hold their current velocities and uncertain radius for a period of time, which is also considered as the system parameters in typical indexing structure such as VCI-tree [11]. So we assume that all the moving object are indexed by a global VCI-tree.

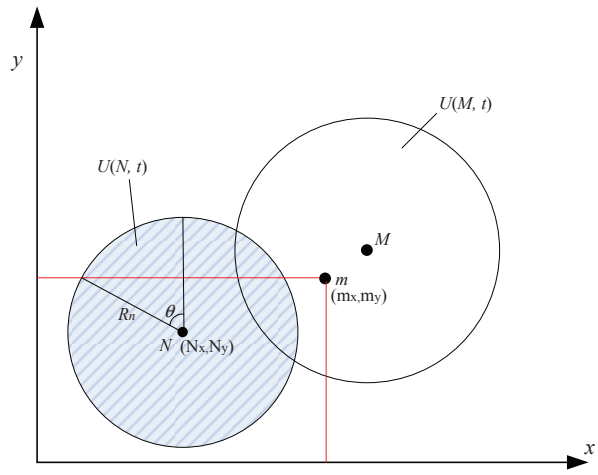


Figure 4. An example of intersection

Figure 5 shows the framework for answering PSQ using the VCI-tree combined with discrete-and-conquer strategy. In particular, the framework consists of five phases, indexing, sampling, bounding, pruning, and refinement. A global VCI-tree is constructed over the entire database first, and then the internal nodes and leaf nodes are uniformly sampled and bounded for pruning purpose. These unqualified nodes will be pruned according to the pruning observations given in Section 3.3, and the left object consist the candidate set. Finally, those object in the candidate set are refined according to the rules given in Section 3.4 to produce the final skyline result.

As shown in the following experimental results, many instances and object can be pruned sharply according to the proposed observations. The final refinement step only has to compute a small portion of instances. Thus, our discrete-and-conquer method has a good scalability on large data sets.

4 PERFORMANCE STUDY

In this section, we conduct several experiments to verify the efficiency and scalability of our proposed approaches. All experiments were conducted on a PC with Intel Pentium Dual-Core 2.5 GHz CPU and 2 GB main memory running Window XP operating system. The programming language used to implement all algorithms is C++.

Without loss of generality, we use the synthetic data sets in independent, correlated and anti-correlated distributions for our experiments. The data sets are obtained from a special data generating framework developed by Brinkhoff [12]. For

Input: The set MO; probability threshold p;
Output: the p-t-skyline in MO;
1: SKY $\neq \emptyset$;
2: build a global VCI-tree GVT for store all $M \in MO$;
3: **for** each internal node $N \in GVT$ **do**
4: select n points uniformly distributed in $U(N, t)$;
5: assign points into layers, compute N_{min} and N_{max};
6: **end for**
7: prune these unqualified internal nodes;
8: extend the remaining nodes into leaf nodes;
9: **for** each leaf node $M \in GVT$ **do**
10: select n points uniformly distributed in $U(M, t)$;
11: assign points into layers, compute M_{min} and M_{max};
12: **end for**
13: prune these unqualified objects within leaf nodes;
14: build a local VCI-tree LVT for all the candidate set;
15: refine the candidate set;
16: **return** SKY;

Figure 5. The framework for PSQ

the synthetic data sets, the domain of each dimension is $[0, 1]$. The dimensionality d is 4 by default. The number of uncertain moving object n is 100 000 by default. We first generate the centers of all uncertain moving object using the data generator. Then, for each uncertain moving object, we use the center to generate an uncertain region where the instances of the object are appear. The radius of the uncertain region follows a normal distribution in range $[0, 0.2]$ with expectation 0.1 and standard deviation 0.05. The instances of the object are distributed uniformly in the region. The maximum velocities of the object follow a uniform distribution with an overall maximum value 0.00005. The instances sampling number of an uncertain moving object s follows a uniform distribution in range $[1, 500]$. Thus, in expectation, each object has 250 instances, and the total number of instances in a data set is 25 000 000 by default. The probability threshold p is 0.3 unless otherwise specified. The skyline query time t is 15 by default after the data sets are generated.

Moreover, to the best of our knowledge, this is the first work to study probabilistic skyline query processing in uncertain moving object databases. The only available method is the exhaustive scan, which sequentially scans the entire database to find out the answers to the query. However, this is very costly. For example, one exhaustive scan of a 50 K 2-D data set with 1KB page size requires about (a floating number takes up four bytes):

$$\frac{10\,\text{ms}}{1\,000\,\text{ms/s}} \times \frac{50\,\text{k} \times 2 \times 4}{1\,\text{k}} \times \frac{50\,\text{k} \times 2 \times 4}{1\,\text{k}} = 1\,600 \text{ seconds}$$

This is much greater than in our method. The detailed query time will be evaluated afterwards.

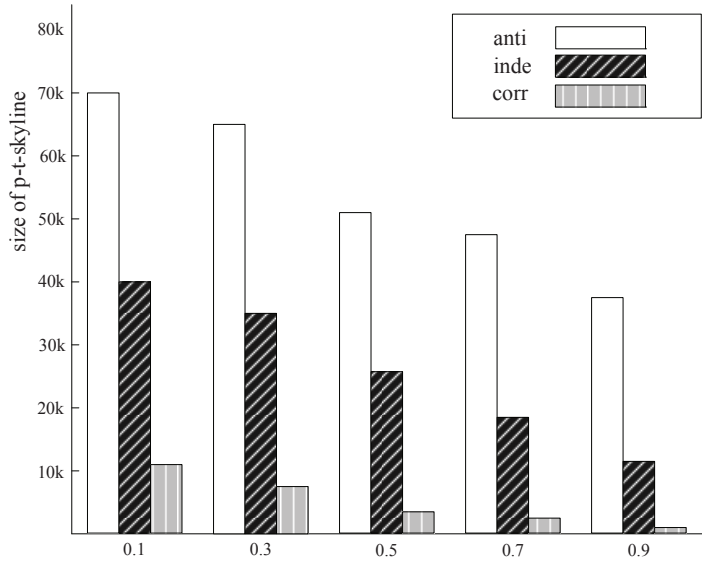
In the first group of experiments, we show the size of probabilistic skylines at query time t (i.e., the number of object in a probabilistic skyline) with respect to three important factors: the probability threshold, the dimensionality and the cardinality. In general, anti-correlated data sets always have the largest skyline size, while correlated data sets have the smallest skyline size. Note that this phenomenon is similar to the situations of skylines on static object.

In particular, the experimental results in Figure 6 a) indicate that the higher the probability threshold, the smaller the skyline size. This is reasonable, since a p'-t-skyline is a subset of a p-t-skyline when $p < p'$. Figures 6 b) and 6 c) show that the skyline size increases with higher dimensionality and larger cardinality, which are also similar to the situations of skylines on precise data sets. Generally speaking, the data set becomes sparser as the dimensionality increases, thus the point has a better chance of not being dominated in all dimensions. On the other hand, as the cardinality increases, more boundary locating object have opportunities not to be dominated. Figure 6 d) shows the skyline size across time, which remains almost constant under little updates. Specially, at each time point, the query algorithm is invoked to re-compute the skyline according to the new positions of object.

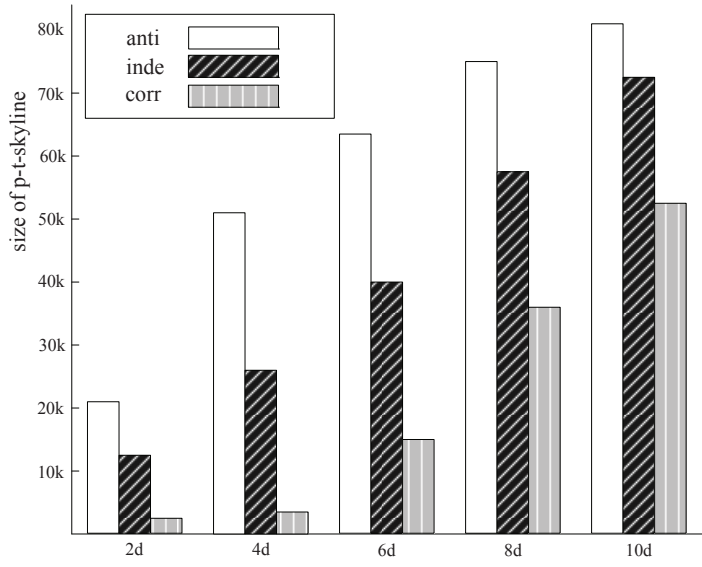
The next group of experiments investigates the runtime of the algorithms with respect to the probability threshold and sampling number. The numbers on the bars as shown in Figure 7 give the exact runtime of algorithms on the data sets. In addition to the discrete-and-conquer (DAC) method described in the paper, for comparison we also made up an exhaustive algorithm (EX) (see above). To compute the p-t-skyline over a data set, without any pruning or validating techniques, EX has to compute the skyline probability for each uncertain moving object. Therefore, it is insensitive to the probability threshold variation and object distribution.

DAC algorithm is much faster than EX for answering skyline query on various data sets. The results in Figure 7 clearly indicate that the pruning techniques in our discrete-and-conquer methods significantly save the cost of computing the exact skyline probabilities of many instances and object. In particular, Figure 7 d) shows the runtime of both algorithms with respect to the instance sampling numbers. EX is insensitive to the variation of s, while DAC initially improves as s increases, but deteriorates as s grows beyond a certain threshold. Note that computing skylines on anti-correlated data set is much more challenging. In the last part of this section, we are focusing on the performance of DAC on anti-correlated data sets.

Figure 8 shows the performance of DAC with respect to dimensionality and cardinality. In particular, we examine the performance of data sets consisted of 100 k uncertain moving object by varying the dimensionality from 2 to 10. Note that the runtime of DAC showed in Figure 8 a) increases when the dimensionality d increases from 2 to 6 but decreases afterwards. The reasons that caused this phenomenon can be concluded into two aspects: On the one hand, the computation cost of dominance checking operation between two instances or object, which is fundamental operation


anti
inde
corr
size of p-t-skyline
80k
70k
60k
50k
40k
30k
20k
10k
0.1
0.3
0.5
0.7
0.9

a)


anti
inde
corr
size of p-t-skyline
80k
70k
60k
50k
40k
30k
20k
10k
2d
4d
6d
8d
10d

b)

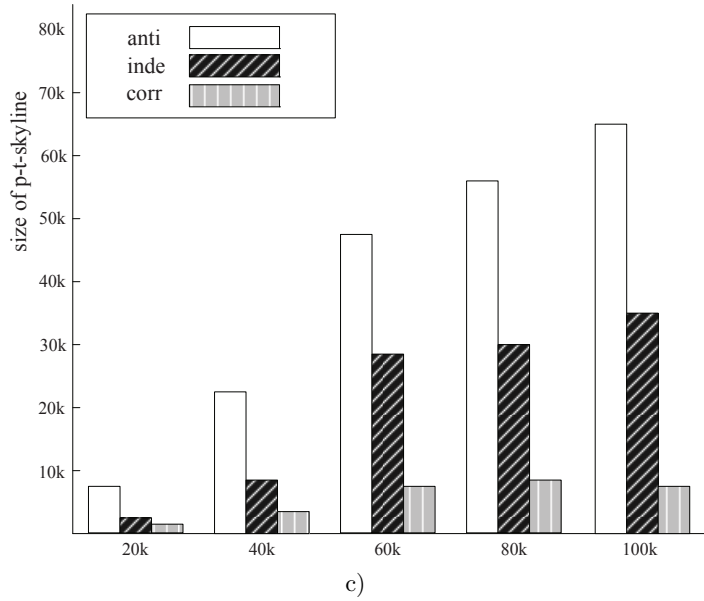
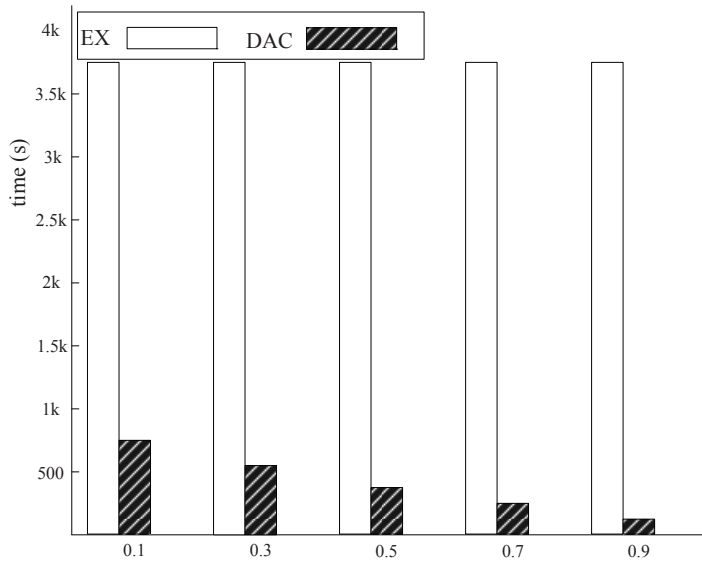
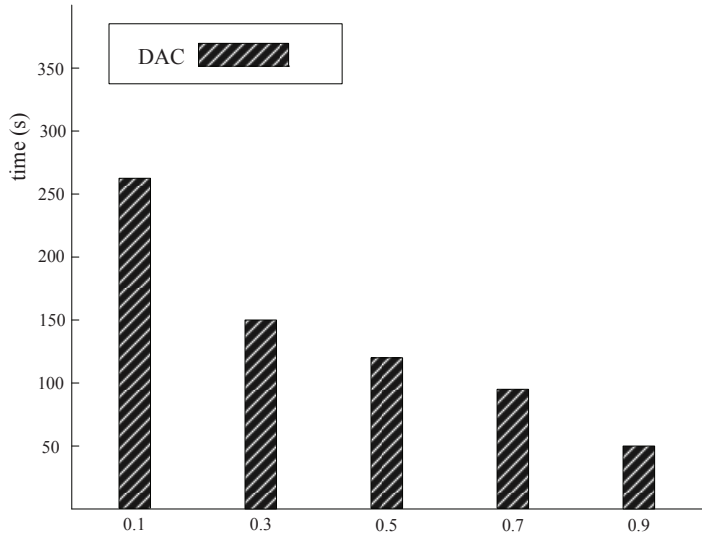


Figure 6. The size of p-t-skyline with respect to probability threshold p, dimensionality d, cardinality m, time t; a) Effect of p, b) Effect of d ($p = 0.5$), c) Effect of m ($p = 0.3$), d) Effect of t ($p = 0.3$, $d = 4$)


EX
DAC
time (s)
4k
3.5k
3k
2.5k
2k
1.5k
1k
500
0.1
0.3
0.5
0.7
0.9

a)


DAC
time (s)
350
300
250
200
150
100
50
0.1
0.3
0.5
0.7
0.9

b)

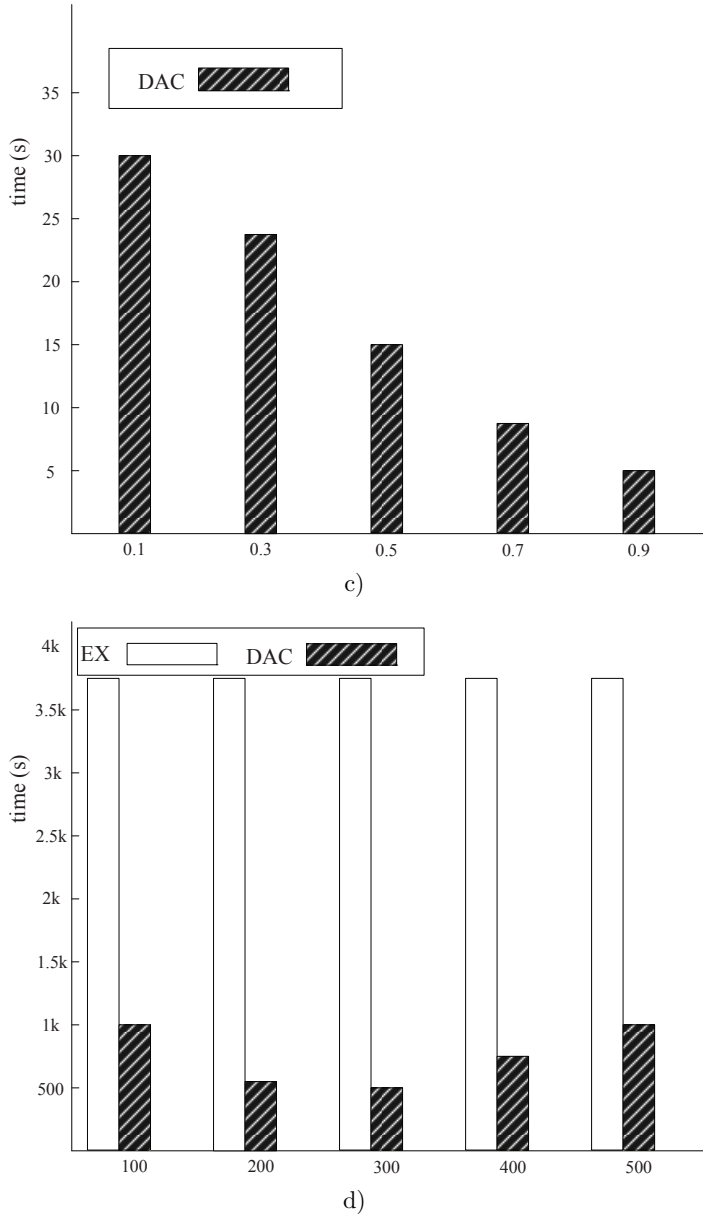


Figure 7. Scalability with respect to probability threshold p, and sampling number s; a) anti-correlated, b) independent, c) correlated, d) effect of s (anti-correlated)

in DAC algorithm, increases as the dimensionality increases; On the other hand, the data set becomes sparser when the dimensionality increases, thus the average numbers of possible dominating object for an uncertain object decreases, which leads to less dominant checking operations. Notice that the trend of runtime reflects a balance of the two factors.

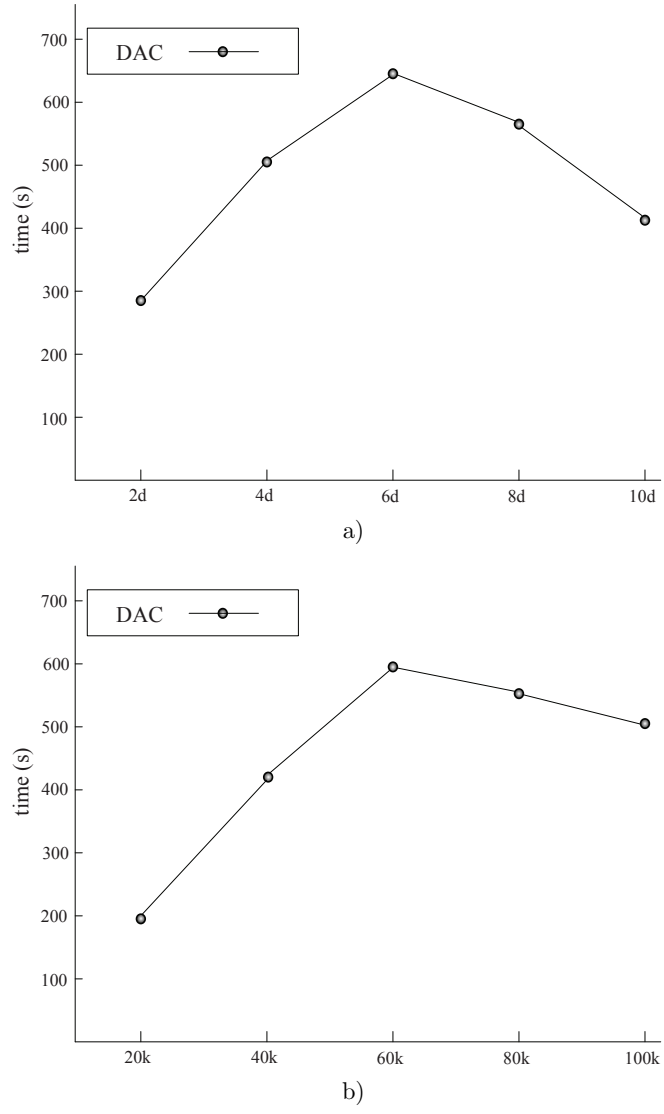


Figure 8. Efficiency with respect to dimensionality d and cardinality m on anti-correlated data sets; a) effect of d (anti-correlated), b) effect of m (anti-correlated)

In order to examine the impact of the cardinality m on DAC algorithm, we examine the performance of data sets with dimensionality equal to 4 and cardinality m varying from 20 k to 100 k. Note that the runtime of DAC showed in Figure 8 b) increases when the cardinality m increases from 20 k to 60 k but decreases afterwards. This is similar to the trend in Figure 8 a). This is reasonable, since the DAC performs better when the data set is sparser. When the cardinality is getting larger, the data set is becoming denser, which leads to more dominant checking operations. On the other hand, in dense data sets, the skyline probability of an instance may improve the probability bounds of more other instances or object, which may lead to less dominant checking operations, thus the DAC performs better too.

In the end, it could be confirmed that our DAC algorithm is scalable according to the dimensionality and the size of the data sets.

5 RELATED WORK

Skyline query has been studied extensively in recent years. Kung et al. [13] proposed the first Skyline algorithm, which is also known as the maximum vector problem. The algorithm is based on the divide-and-conquer strategy (DC), which divides the database into several sub-regions, calculates the skyline in each sub-region, and produces the final skyline from the points in the regional skylines. Börzsönyi et al. [3] introduced the first skyline operator to the database community and proposed the block-nested-loop algorithm (BNL) together with the extended DC algorithm. Both BNL and DC compare each point within the entire database with the others, and report it as a skyline point only if there is no other object which can dominate it. In order to progressively report the skyline points, based on the rationale of BNL, sort-filter-skyline algorithm (SFS) [5] improves the performance by first sorting the data according to a monotone function. A bitmap-based method [9] converts each point p to a bit vector, which encodes the number of points having a smaller coordinate than p on every dimension. The skyline is then obtained using only bit operations. However, bitmaps can be extremely large for large values, and it cannot guarantee a quick response.

As opposed to the above algorithms that must read the whole database at least once, index-based methods need to visit only a fraction of the dataset. An extension of B-trees algorithm [9] organizes the high dimensional data in the form of single dimensional space where object are clustered according to their coordinates within each dimension. Thus, the skyline points can be determined without checking those points not accessed yet. However, the B-tree algorithm cannot return the skyline points according to the preference. Kossmann et al. [4] proposed an index based progressive processing algorithm NN (nearest-neighbor), which is based on the depth-first nearest search via R*-tree. The performance of NN deteriorates as the dimensionality increases. For further improvement, Papadias et al. [8] proposed the branch-and-bound skyline (BBS), based on the best-first nearest neighbor search.

It has been proved that BBS is I/O optimal as it accesses fewer disk pages than any algorithm based on R-tree.

Yuan et al. [14] propose the sky-cube, which consists of the skylines in all possible sub-spaces. Lin et al. [1] investigate continuous skyline monitoring on data streams. Recently, Lian et al. [19] have studied efficient and accurate query processing of reverse skyline query over uncertain data, considering both monochromatic and bichromatic situations. Ding et al. [20] consider distributed probabilistic skyline over uncertain data. Pei et al. [2] tackle the problem of skyline analysis on uncertain data and Huang et al. [6] introduce the continuous skyline over precise moving data. However, the above algorithms are not applicable in a mobile environment in which properties like distance are functions that take the current position as a parameter. The topic of this paper differs from [2] and [6] in that we aim at computing the skyline of uncertain moving object, as opposed to static uncertain object and precise moving object. Most recently, based on the past uncertain location and velocity information, Zhang et al. [21] present techniques that enable inference of the current and future uncertain locations efficiently. Furthermore, an adapted B^X-tree structure is invented to index the uncertain moving object.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we extended the well-known skyline analysis to uncertain moving object, and introduced a novel concept of probabilistic skyline queries. We studied the probabilistic properties of the solution to these queries. We analytically proved that a set of probabilistic skyline object exists. We also proved that the instances of the object can be used to prune or validate the other uncertain moving object at a certain time. Based on several proposed observations, we developed efficacious algorithm to tackle the problem of computing probabilistic skylines on uncertain moving data. Using synthetic data sets, we illustrated the efficiency and scalability of our proposed algorithm.

Instead of re-computing the skyline when any of the object change their positions, we are planning to investigate the connection between the spatial positions of moving object and their dominance relationship, which may find changes in the skyline result and reduce much computation cost in the result maintenance. We also plan to use the continuous nearest neighbor techniques to solve skyline queries over uncertain moving object in the future. It is also interesting to investigate how the NN algorithm can be combined with discrete-and-conquer strategy for further performance improvement. For instance, it would be possible to use nearest neighbor algorithm partition the data space and then use the DAC algorithm to process each region. Such a hybrid approach would give a possible result quickly and continue to output the skyline efficiently.

Acknowledgment

This work was supported by the National Natural Science foundation of China under Grant No. 61100060, the China Postdoctoral Science Foundation funded project under grant No. 20100471179, the Natural Science foundation of Hubei Province under Grant No. 2011CDB037 and the Central Colleges of basic scientific research and operational costs under grant No. 2011QN054.

REFERENCES

[1] Lin, X. Y.—Yuan, Y. D.—Wang, W.—Lu, H.: Stabbing the Sky: Efficient Skyline Computation over Sliding Windows. Proceedings of the 21st International Conference on Data Engineering, ICDE '05, Tokyo, April 2005, pp. 502–513.

[2] Pei, J.—Jiang, B.—Lin, X.—Yuan, Y. D.: Probabilistic Skylines on Uncertain Data. Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB '07, Vienna, September 2007.

[3] Börzsönyi, S.—Kossmann, D.—Stocker, K.: The Skyline Operator. Proceedings of the 17th International Conference on Data Engineering, ICDE '01, Heidelberg, April 2001, pp. 421–430.

[4] Kossmann, D.—Ramsak, F.—Rost, S.: Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. Proceedings of 28th International Conference on Very Large Data Bases, VLDB '02, Hong Kong, August 2002, pp. 275–286.

[5] Chomicki, J.—Godfrey, P.—Gryz, J.—Liang. D.: Skyline with Presorting. In: U. Dayal, K. Ramamritham, and T. M. Vijayaraman (Eds.): IEEE Computer Society, Proceedings of the 19th International Conference on Data Engineering, ICDE '03, Bangalore, March 2003, pp. 717–816.

[6] Huang Z. Y.—Lu, H.—Beng, C. O.—Anthony. T.: Continuous Skyline Queries for Moving Objects. IEEE Trans. Knowledge and Data Engineering, Vol. 18, 2006, No. 12, pp. 1645–1658.

[7] Sharifzadeh, M.—Shahabi, C.: The Spatial Skyline Queries. In: U. Dayal, K. Y. Whang, D. B. Lomet, et al. (Eds.): ACM 2006, Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB '06, Seoul, September 2006, pp. 751–762.

[8] Papadias, D.—Tao, Y.—Fu, G.—Seeger, B.: Progressive Skyline Computation in Database Systems. ACM Trans. Database Syst., Vol. 30, 2005, No. 1, pp. 41–82.

[9] Tan, K. L.—Eng, P. K.—Beng, C. O.: Efficient Progressive Skyline Computation. Proceedings of 27th International Conference on Very Large Data Bases, VLDB '01, Roma, September 2001, pp. 301–310.

[10] Huang, Z.—Jensen, C. S.—Lu, H.—Beng, C. O.: Skyline Queries Against Mobile Lightweight Devices in MANETs. In: Ling Liu, Andreas Reuter, Kyu-Young Whang, Jianjun Zhang (Eds.): IEEE Computer Society 2006, Proceedings of the 22nd International Conference on Data Engineering, ICDE '06, Atlanta, April 2006.

[11] Prabhakar, S.—Xia, Y.—Kalashnikov, D.—Aref, W.—Hambrusch, S.: Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects. IEEE Trans. Computers, Vol. 51, 2002, No. 10, pp. 1124–1140.

[12] Brinkhoff, T.: A Framework for Generating Network Based Moving Objects. Geoinformatica, Vol. 2, 2002, No. 6, pp. 153–180.

[13] Kung, H. T.—Luccio, F.—Preparata, F.: On Finding the Maxima of a set of Vectors. Journal of the ACM, Vol. 22, 1975, No. 4.

[14] Yuan, Y. D.—Lin, X.—Liu, Q.—Wang, W.—Yu, J. X.—Zhang, Q.: Efficient Computation of the Skyline Cube. Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, VLDB '05, Norway, August 2005, pp. 241–252.

[15] Cheng, R.—Dmitri, V. K.—Prabhakar, S.: Querying Imprecise Data in Moving Object Environments. In IEEE Transactions on Knowledge and Data Engineering, Vol. 16, 2004, No. 9, pp. 1112–1127.

[16] Cheng, R.—Vitter, J. S.: Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data. In: M. A. Nascimento, M. T. Özsu, D. Kossmann, et al. (Eds.): Morgan Kaufmann 2004, Proceedings of the 30th International Conference on Very Large Data Bases, VLDB '04, Toronto, August 2004, pp. 876–887.

[17] Tao, Y. F.—Cheng, R.—Xiao, X. K.—Ngai, W. K.—Kao, B.—Prabhakar, S.: Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions. Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, VLDB '05, Norway, August 2005.

[18] Ding, X. F.—Lu, Y. S.: Indexing the Imprecise Positions of Moving Objects. Proceedings of the ACM SIGMOD 2007 Ph. D. Workshop on Innovative Database Research, Beijing, June 2007, pp. 45–52.

[19] Lian, X.—Chen, L.: Monochromatic and Bichromatic Reverse Skyline Search over Uncertain Database. In: Jason Tsong-Li Wang (Ed.): ACM 2008, Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD '08, Vancouver, June 2008, pp. 213–226.

[20] Ding, X. F.—Jin, H.: Efficient and Progressive Algorithms for Distributed Skyline Queries over Uncertain Data. IEEE Transactions on Knowledge and Data Engineering, Vol. 24, 2012, No. 8, pp. 1448–1462.

[21] Zhang, M. H.—Chen, S.—Jensen, C. S.—Beng, C. O.—Zhang, Z. J.: Effectively Indexing Uncertain Moving Objects for Predictive Queries. Proceedings of the 35th International Conference on Very Large Data Bases, VLDB '09, Lyon, August 2009, pp. 261–272.

[22] Trajcevski, G.—Tamassia, R.—Ding, H.—Scheuermann, P.—Cruz, I. F.: Continuous Probabilistic Nearest-Neighbor Queries for Uncertain Trajectories. In: M. L. Kersten, B. Novikov, J. Teubner, V. Polutin, and S. Manegold (Eds.): ACM International Conference Proceeding Series 360 ACM 2009, Proceedings of the 12th International Conference on Extending Database Technology, EDBT '09, Saint Petersburg, March 2009, pp. 874–885.

Xiaofeng Ding received the Ph. D. degree in computer science in 2009 from the Huazhong University of Science and Technology. Currently, he is an Assistant Professor in the School of Computer Science and Technology at Huazhong University of Science and Technology. His research interests include distributed computing, query processing, data privacy, uncertain data management, peer-to-peer computing, and spatial databases.

Hai Jin received the Ph. D. degree in computer engineering in 1994 from Huazhong University of Science and Technology (HUST), China, where he is currently the Cheung Kong Professor and the Dean of the School of Computer Science and Technology. In 1996, he was awarded a German Academic Exchange Service Fellowship to visit the Technical University of Chemnitz in Germany. He worked at the University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded the Distinguished Young Scholar Award from the National Science Foundation of China in 2001. He is the chief scientist of ChinaGrid, the largest grid computing project in China. He is a member of Grid Forum Steering Group (GFSG). He has co-authored 15 books and published more than 400 research papers. His research interests include distributed computing, computer architecture, virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, and network security. He is a senior member of the IEEE and a member of the ACM.

Hui Xu received her B. SC. degree in computer science from Zhengzhou University, Zhengzhou, People's Republic of China, in 2010. She is now Ph. D. candidate in the School of Computer Science and Technology at Huazhong University of Science and Technology. Her research interests include query processing, sequence alignment, and uncertain data management.

Wei Song received his B. Sc. degree in computer science and technology from South-Central University for Nationalities, Wuhan, People's Republic of China in 2004, his M. Sc. and Ph. D. degrees in information and communication engineering from Chonbuk National University, Jeonbuk, Korea in 2006 and 2009. He is now Associate Professor at Jiangnan University. His research interests include pattern recognition, data mining, information retrieval, evolutionary computation, neural networks and artificial intelligence.