Stochastic Skylines

WENJIE ZHANG, The University of New South Wales
XUEMIN LIN, East China Normal University and The University of New South Wales
YING ZHANG and MUHAMMAD AAMIR CHEEMA, The University of New South Wales
QING ZHANG, The Australian EHealth Research Centre, CSIRO

In many applications involving multiple criteria optimal decision making, users may often want to make a personal trade-off among all optimal solutions for selecting one object that fits best their personal needs. As a key feature, the skyline in a multidimensional space provides the minimum set of candidates for such purposes by removing all points not preferred by any (monotonic) utility/scoring functions; that is, the skyline removes all objects not preferred by any user no matter how their preferences vary. Driven by many recent applications with uncertain data, the probabilistic skyline model is proposed to retrieve uncertain objects based on skyline probabilities. Nevertheless, skyline probabilities cannot capture the preferences of monotonic utility functions. Motivated by this, in this article we propose a novel skyline operator, namely stochastic skylines. In the light of the expected utility principle, stochastic skylines guarantee to provide the minimum set of candidates to optimal solutions over a family of utility functions. We first propose the *lskyline* operator based on the *lower orthant orders*. lskyline guarantees to provide the minimum set of candidates to the optimal solutions for the family of monotonic multiplicative utility functions. While lskyline works very effectively for the family of multiplicative functions, it may miss optimal solutions for other utility /scoring functions (e.g., linear functions). To resolve this, we also propose a general stochastic skyline operator, *gskyline*, based on the *usual orders*. gskyline provides the minimum candidate set to the optimal solutions for all monotonic functions. For the first time regarding the existing literature, we investigate the complexities of determining a stochastic order between two uncertain objects whose probability distributions are described *discretely*. We firstly show that determining the lower orthant order is NP-complete with respect to the dimensionality; consequently the problem of computing lskyline is NP-complete. We also show an interesting result as follows. While the usual order involves more complicated geometric forms than the lower orthant order, the usual order may be determined in polynomial time regarding all the inputs, including the dimensionality; this implies that gskyline can be computed in polynomial time. A general framework is developed for efficiently and effectively retrieving lskyline and gskyline from a set of uncertain objects, respectively, together with efficient and effective filtering techniques. Novel and efficient verification algorithms are developed to efficiently compute lskyline over multidimensional uncertain data, which run in polynomial time if the dimensionality is fixed, and to efficiently compute gskyline in polynomial time regarding all inputs. We also show, by theoretical analysis and experiments, that the sizes of lskyline and gskyline are both quite similar to that of conventional skyline over certain data. Comprehensive experiments demonstrate that our techniques are efficient and scalable regarding both CPU and IO costs.

W. Zhang was partially supported by ARC DE120102144 and DP120104168. X. Lin was partially supported by ARC DP0987557, ARC DP110102937, ARC DP120104168, and NSFC61021004. Y. Zhang was partially supported by DP110104880 and UNSW ECR grant PSE1799.
Authors' addresses: W. Zhang, School of Computer Science and Engineering, The University of New South Wales, Australia; email: zhangw@cse.unsw.edu.au; X. Lin (corresponding author), East China Normal University, China and the University of New South Wales, Australia; Y. Zhang and M. A. Cheema, School of Computer Science and Engineering, The University of New South Wales, Australia; Q. Zhang, The Australian EHealth Research Centre, CSIRO, Australia.

DOI 10.1145/2188349.2188356 http://doi.acm.org/10.1145/2188349.2188356

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process; clustering*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Skyline, uncertain data, stochastic order

ACM Reference Format:
Zhang, W., Lin, X., Zhang, Y., Cheema, M. A., and Zhang, Q. 2012. Stochastic skylines. ACM Trans. Datab. Syst. 37, 2, Article 14 (May 2012), 34 pages.
DOI = 10.1145/2188349.2188356 http://doi.acm.org/10.1145/2188349.2188356

1. INTRODUCTION

In a d-dimensional space R^d, the *skyline* is defined over given preferences of coordinate values on each dimension (i.e., either smaller or larger coordinate values are preferred). Without loss of generality, in the rest of the article we assume that smaller coordinate values are preferred on each dimension and all points are in R^d_+ (i.e., coordinate values are nonnegative). Given two points x and y in R^d_+, x *dominates* y if x is not greater than y on each dimension and is smaller than y on one dimension. Given a set D of objects (points) in R^d_+, the skyline of D consists of the points in D which are not dominated by any other points in D.

When users want to select a top object (point) from a set D of objects (points) in R^d_+ based on their preference of smaller values on each dimension, a point in D that dominates every other point in D would be the best choice. Nevertheless, D may not always contain such a point. Therefore, scoring functions are often required to rank the points in D to generate-optimal solution. To capture the preference of smaller values on each dimension and make the optimal solution with the maximum score, a scoring function required to rank objects in D should be *decreasing* (see Section 2.2 for the precise definition). Clearly, a single scoring function on a point may be easily affected by an outlier coordinate value of the point. Consequently, users may not be content with the optimal solution based on a single decreasing scoring function and may highly anticipate to make a personal trade-off to select one object which may best fit their personal needs among the set S of points with the property that for each point $y \in D - S$, there is a point $x \in S$ such that $f(x) \geq f(y)$ for any decreasing function; we call such an S a *superior set* of D. Clearly, a superior set S in D contains the (score-based) optimal solutions over all possible decreasing scoring functions.

It is well-known [Börzsönyi et al. 2001; Steuer 1995] that x dominates y if and only if for any decreasing function f, $f(x) \geq f(y)$; this is formally stated in page 266 of Shaked and Shanthikumar [2007] if a point is regarded as an uncertain object with only one instance that has the occurrence probability 1. Consequently, the skyline S of D provides the superior set of D with the minimum size for users to make their personal trade-offs. Such a personal trade-off is very effective especially when the number of skyline points is small. For this reason, skyline computation over certain data has received significant research attention in the last decade (e.g., Börzsönyi et al. [2001], Kossmann et al. [2002], Papadias et al. [2003], Tan et al. [2001], and Zhang et al. [2009]).

Expected Utility Principle. Ranking uncertain objects against multiple criteria has a long history in economics, finance, and mathematics; see Levy [1992], Shaked and Shanthikumar [2007], and Kijima and Ohnishi [1999] for example. The *expected utility principle* is the most popular model [Levy 1992; Kijima and Ohnishi 1999] to select the optimal uncertain object against multiple criteria. In the light of the expected utility principle, given a function f, an uncertain object U with the maximum expected utility is the optimal solution; that is, select U to maximize $E[f(U)]$ for a given utility function f.

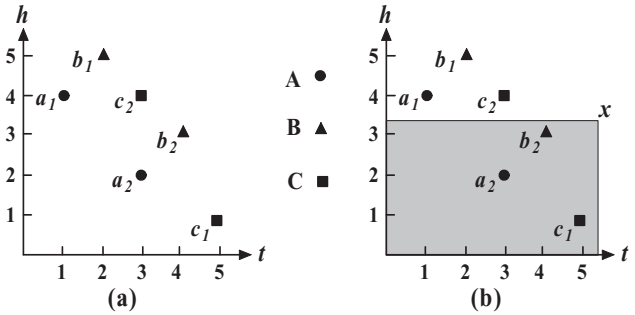

Fig. 1. Motivating example for lskyline.

While the expected utility principle is widely believed the best scoring model when the the world is rational, a single utility function to score an uncertain object U may also be easily affected by outliers, including the outliers in the probability distributions of U. To resolve this, in this article we propose to investigate the problem of efficiently computing the *stochastic-order-based skyline*, following the aforesaid motivation of skyline.

Stochastic Order. Stochastic orders have been effectively used in many real-life applications [Levy 1992; Shaked and Shanthikumar 2007; Kijima and Ohnishi 1999], including economics, finance, and multicriteria statistic decision making to facilitate the expected utility principle. Generally, a *stochastic order* may be defined over any family $\mathcal{F}$ of functions as follows. Given a family $\mathcal{F}$ of utility (scoring) functions from all users, an uncertain object (random variable) U *stochastically dominates* V regarding $\mathcal{F}$, denoted by $U \prec_{\mathcal{F}} V$ if and only if $E[f(U)] \geq E[f(V)]$ for each $f \in \mathcal{F}$ (see Kijima and Ohnishi [1999]); that is, all users prefer U to V according to the expected utility principle.

Stochastic Skylines. Given a set $\mathcal{U}$ of uncertain objects, the stochastic-order-based skyline, namely stochastic skylines regarding $\mathcal{F}$, is the subset of $\mathcal{U}$ such that each object U in the stochastic skylines is not stochastically dominated by any other object in $\mathcal{U}$ regarding $\mathcal{F}$. Based on the preceding definition of stochastic orders, it is immediate that the stochastic skylines regarding $\mathcal{F}$ provide the minimum set of candidates to the optimal solutions (maximum values) for all functions in $\mathcal{F}$ by removing the objects not preferred by any function in $\mathcal{F}$. Therefore, stochastic skylines provide the superior set of objects in $\mathcal{U}$, regarding $\mathcal{F}$ and the expected utility principle, with the minimum size for users to make a personal trade-off to choose an object which fits best their individual needs; thus, stochastic skylines regrading $\mathcal{F}$ are robust regarding outliers.

Several stochastic orders have been defined and their mathematical properties have been well studied in the statistics literature [Shaked and Shanthikumar 2007; Kijima and Ohnishi 1999]. Multiplicative nonnegative decreasing functions are a very popular family $\mathcal{F}$ of scoring functions to rank an object in R^d_+, that is, $\mathcal{F} = \{\prod_{i=1}^{d} f_i(x_i)\}$ where each f_i (for $1 \leq i \leq d$) is nonnegative decreasing. In this article, we first investigate the problem of efficiently computing the stochastic skylines regarding *lower orthant order* [Shaked and Shanthikumar 2007], namely lskyline. Intuitively, an uncertain object U stochastically dominates V regarding the lower orthant order if and only if the probability mass of U is not smaller than the probability mass of V in any rectangular region with the origin as the lower corner (e.g., the shaded region in Figure 1(b) and Figure 2(a)). It is shown [Shaked and Shanthikumar 2007] that the lower orthant order $\prec_{\mathcal{F}}$ is the stochastic order defined over the family $\mathcal{F}$ of multiplicative decreasing functions (see Section 2 for details); consequently, lskyline gives the superior set of

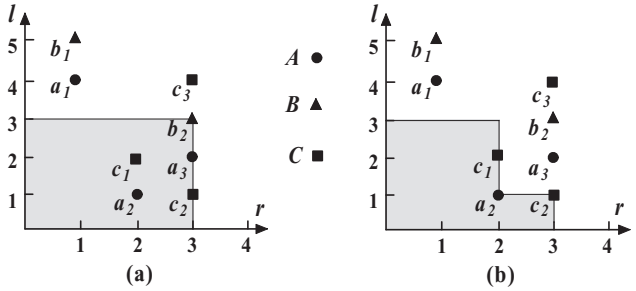

Fig. 2. Motivating example for gskyline.

objects in a set $\mathcal{U}$ of uncertain objects, regarding the family of all negative multiplicative decreasing functions and the expected utility principle, with the minimum size.

While lskyline works effectively for the family of multiplicative functions, it may miss optimal solutions for other utility/scoring functions (say, linear functions). Motivated by this, in this article we also study the problem of efficiently computing the stochastic skyline based on the usual order [Shaked and Shanthikumar 2007], called "general stochastic skyline" (gskyline). Intuitively, U dominates V regarding the usual order if and only if the probability mass of U is not smaller than the probability mass of V in any (possibly infinite) combination of the rectangular regions with the origin as the lower corner (e.g., the shaded regions in Figure 2(a) and (b)). As shown in Shaked and Shanthikumar [2007] and Section 2.2, the usual order is the stochastic order over the family of all decreasing functions. Therefore, gskyline gives the superior set of objects in a set $\mathcal{U}$ of uncertain objects, regarding the family of all decreasing functions and the expected utility principle, with the minimum size.

pskyline vs lskyline. Driven by many recent applications involving uncertain data (e.g., environment surveillance, market analysis, WWW, and sensor networks), research in uncertain data management has drawn considerable attention from the database community. The skyline analysis over uncertain data has been firstly proposed in Pei et al. [2007] where the *possible world* semantics is adopted to calculate the probabilities, namely *skyline probabilities*, of uncertain objects not being dominated by other uncertain objects. In Pei et al. [2007], efficient techniques are developed to retrieve uncertain objects with skyline probabilities greater than a given threshold, namely, *pskyline*, while Atallah et al. [2011] provides efficient techniques to compute skyline probabilities for all objects. While effectively capturing the possible dominance relationships among uncertain objects, skyline probabilities do not provide the candidature of optimal solutions regarding the expected utility principle; this is illustrated by the following example.

Assume that a head coach wants to select the best high-jump athlete from all athletes in her team to attend an international game and decides to evaluate the athletes against their game performances in the recent years, say the last 3 years, where the performance of each athlete in a game is recorded by the final height h and the total number t of failed trials over all attempted heights before successfully passing the final height (h, t). To conform with the preference of smaller values, assume that h is recorded into 5 *bands* instead of actual value: band 1: within a reach of smashing the current world record, band 2: world leading heights, band 3: good, band 4: fair, band 5: poor. The coach wants to select a player who is very stable (i.e., t is minimized) and jumps high (i.e., h is minimized). Clearly, these two criteria might conflict with each other. Consequently, a utility function $f = f_1(h) \times f_2(t)$ may be employed by the coach to evaluate the overall performance of a player in a game, where f_1 and

f_2 are nonnegative decreasing functions, mapping $[1, 5]$ to $[0, 1]$ and $[0, \infty)$ to $[0, 1]$, respectively, with $f_1(1) = 1$ and $f_2(0) = 1$. The coach selects the athlete with the maximum value of $(f_1(h) \times f_2(t))$. Nevertheless, the game performance of an athlete may fluctuate from game to game due to various reasons. Therefore, it is important to evaluate players against their game performance statistic distributions. For this purpose, an athlete U may be treated as an uncertain object and her performance in each past game may be treated as an instance in a 2-dimensional space with the same probability to occur if no other information is available. The coach could select a player U such that $E(f(U))$ is maximized. Figure 1 gives a small-scale example where 3 players are involved.

As depicted in Figure 1(a), suppose that A, B, and C have 2 instances, respectively. Each instance in objects A and B has the occurrence probability $\frac{1}{2}$. The occurrence probabilities of c_1 and c_2 are $\frac{1}{100}$ and $\frac{99}{100}$, respectively, assuming that the player C has the same performance c_2 for 99 games out of 100. While choosing an object U from A, B, and C to maximize $E[f(U)]$ in the light of the *expected utility principle*, $E[f(A)] = \frac{1}{2} f_1(4) f_2(1) + \frac{1}{2} f_1(2) f_2(3)$, $E[f(B)] = \frac{1}{2} f_1(5) f_2(2) + \frac{1}{2} f_1(3) f_2(4)$, and $E[f(C)] = \frac{1}{100} f_1(1) f_2(5) + \frac{99}{100} f_1(4) f_2(3)$. Note that nonnegative decreasing utility functions f_1 and f_2 could be in any form depending on what kind of risks and trade-offs the coach wants to take; for instance, f_1 and f_2 could be in exponential forms such as $f_1(h) = e^{a(1-h)}$ and $f_2(t) = e^{-bt}$ where $a > 0$ and $b > 0$ may personally weigh the importance of h and t.

Nevertheless, A is always preferred to B since $E[f(A)] \geq E[f(B)]$ for any nonnegative decreasing functions f_1 and f_2, respectively. Suppose that $f(h, t) = f_1(h) f_2(t)$ where $f_1(h) = e^{1-h}$ and $f_2(t) = e^{-t}$. It can be immediately verified that $E[f(A)] > E[f(C)]$. If the coach is taking more risks and only wants to select the athlete with a chance to smash the world record, another nonnegative multiplicative decreasing function $f'(h, t) = f'_1(h) f'_2(t)$ may be chosen such that f_1 can be defined such that $f'_1(1) = 1$ and $f'_1(h) = 0$ if $h > 1$ and f'_2 still uses e^{-t}. If f' is used, then C is the optimal solution since $E[f'(A)]$ and $E[f'(B)]$ are 0. The example shows that A is preferred to B by any users (i.e., by any such multiplicative decreasing utility functions) and should be excluded as a candidate to any optimal solutions, while A and C should be kept as candidates. It is immediate that lskyline serves for this purpose; that is, lskyline contains A and C but excludes B.

While very useful in applications to determine probabilistic dominance relationships, skyline probabilities cannot capture the preferences of utility (scoring) functions regarding the expected utility principle. Regarding the example in Figure 1(a), it can be immediately verified that the skyline probability of A is 1, the skyline probability of B is $\frac{1}{2}$, and the skyline probability of C is $\frac{1}{100}$. Clearly, if we choose atheteses based on skyline probability values, then B is always preferred to C; that is, there is no chance to exclude B without excluding the object C. This is an inherent limitation of the probabilistic skyline model.

Why gskyline? In what follows we show by an example that lskyline may miss the optimal solutions for other utility/scoring functions (say, linear functions).

Assume the head coach in another team wants to select the best high-jump athlete against game performance regarding the final rank r of an athlete in a game and the rank l of the game; that is, each game performance of an athlete is recorded by (r, l) where the lowest r and l values are 1 (the lower values, the better). Clearly, it is desirable to select an athlete who always has the best performance in a top tournament; that is, prefer lowest r and l. Similarly, r and l might often conflict with each other; thus, a utility function $f(r, l)$ may be employed to evaluate the overall performance of an athlete in a game where f is nonnegative decreasing, mapping $[1, \infty) \times [1, \infty)$ to

[0, 1] with $f(1, 1) = 1$. For the same reason as before, an athlete may be treated as an uncertain object U and her each game performance is treated as an instance in R^2_+ with the same occurrence probability if there is no specific information. The coach may select an athlete U with the maximal value of $E[f(U)]$ based on the expected utility principle. Figure 2(a) gives a small-scale example with 3 athletes A, B, and C.

In Figure 2, assume that A has 3 instances a_1, a_2, and a_3 with the occurrence probabilities $\frac{1}{2}$, $\frac{1}{7}$, $\frac{5}{14}$, respectively, assuming the game performance a_1 occurs 7 times, a_2 occurs 2 times, and a_3 occurs 5 times. Suppose that B has 2 instances b_1 and b_2 with the occurrence probabilities $\frac{1}{2}$ and $\frac{1}{2}$, respectively; and C has 3 instances c_1, c_2, and c_3 with the occurrence probabilities $\frac{1}{8}$, $\frac{1}{8}$, and $\frac{3}{4}$ respectively. Immediately, the expected utility of A regarding a utility function f is $E[f(A)] = \frac{1}{2}f(a_1) + \frac{1}{7}f(a_2) + \frac{5}{14}f(a_3) = \frac{1}{2}f(1, 4) + \frac{1}{7}f(2, 1) + \frac{5}{14}f(3, 2)$. Similarly, we can present $E[f(B)]$ and $E[f(C)]$.

A nonnegative decreasing utility function f may be in any form subject to the levels of risks and trade-offs that the coach wants to take. For example, if the coach wants to focus on the top 3 performances in ranks 1, 2, and 3 tournaments, f may be defined as $f(r, l) = \frac{1}{2}\frac{4-r}{3} + \frac{1}{2}\frac{4-l}{3}$ when $r \leq 3$ and $l \leq 3$ (the shaded area in Figure 2(a)) and 0 otherwise. It can be immediately verified that $E[f(A)] = \frac{25}{84}$ is the maximum among the 3 objects; that is, A is optimal. If the coach wants to only count the top 3 performances in rank 1 tournaments and the top 2 performances in rank 2 and 3 tournaments, f may be defined as $f(r, l) = \frac{1}{2}\frac{4-r}{3} + \frac{1}{2}\frac{4-l}{3}$ in the shaded area of Figure 2(b) and 0 otherwise; in this case C is optimal since $E[f(C)] = \frac{1}{6}$ is the maximum.

The example shows that A and C should be kept as candidates for users to make a personal trade-off since A and C will be the optimal solutions for some nonnegative decreasing utility functions, respectively, regarding the expected utility principle. It can be immediately verified that for any nonnegative decreasing function f, $E[f(A)] \geq E[f(B)]$; that is, A is always preferred to (better than) B. Consequently, B should be *excluded* as a candidate for users to make a personal trade-off. These can be exactly captured by gskyline.

Moreover, it can be immediately verified that the probability mass of A is not smaller than the probability mass of C in any rectangular region with the origin as the lower corner (e.g., the shaded region in Figure 2(a)); that is, A dominates C regarding the lower orthant order. Thus, lskyline excludes C, and lskyline may miss the optimal solutions for some utility/scoring functions. It can also be immediately verified that the skyline probabilities of A, B, and C are $\frac{51}{56}$, $\frac{7}{16}$, and $\frac{3}{14}$, respectively. Clearly, B is always preferred to C based on the skyline probability values; that is, it is impossible to exclude the object B without excluding the object C based on skyline probabilities; this demonstrates again that skyline probabilities do not guarantee candidatures of optimal solutions regarding the expected utility principle.

Challenges and Contributions. While the two small-scale examples given earlier illustrate the benefits of computing stochastic skylines, lskyline and gskyline are useful tools in any application requiring to rank uncertain objects in the light of expected utility principle. They also demonstrate that lskyline suits for the applications where only nonnegative decreasing multiplicative ranking functions are involved, and gskyline suits for the applications where all monotonic ranking functions are involved. Motivated by these, in the article we study the problem of efficiently computing both lskyline and gskyline. Consider that in many applications, the probability distribution of an uncertain object is described by a set of instances and their occurrence probabilities (i.e., discrete cases). In the article, our investigation focuses on discrete cases of the probability distributions of objects.

The main challenge for computing stochastic skylines is to efficiently check the stochastic dominance relationship between two uncertain objects U and V regarding the lower orthant order and usual order, respectively. A naive way to check the stochastic dominance relationship between two uncertain objects U and V regarding the lower orthant order is to enumerate all rectangular regions with the origin as the lower corner to determine the lower orthant order between two objects; this involves testing an infinite number of regions. Enumerating all the combinations of such rectangular regions to determine usual order only incurs more costs. While testing the stochastic dominance relationship between two uncertain objects U and V regarding the lower orthant order and usual order, respectively, can be easily reduced to the finite computation, trivially conducting such computation may involve exponential time. To the best of our knowledge, this is the first attempt to introduce the stochastic skylines model over uncertain data. Our principal contributions can be summarized as follows.

—We introduce a novel stochastic skyline model on uncertain data with the aim to provide a minimum set of candidates to the optimal solutions for users to make their personal trade-offs over the family of multiplicative decreasing scoring functions and the family of general decreasing scoring functions, respectively.
—We are the first to show that the problem of determining the lower orthant order between two objects is NP-complete regarding the dimensionality. A novel, efficient algorithm is developed to verify if an uncertain object is stochastically dominated by another object based on the lower orthant order. The algorithm runs in polynomial time if the dimensionality is fixed.
—We are the first to show that the problem of testing the usual order can be solved in polynomial time regarding all inputs, including the dimensionality, though it involves more complex geometric forms in the definition than that in the definition of the lower orthant order. We convert the problem to the max-flow problem, and then propose novel techniques to effectively reduce the size of auxiliary networks to significantly speed up the testing.
—To efficiently compute stochastic skylines among a large number of objects, following the *filtering verification* paradigm we develop novel spatial- and statistics-based filtering techniques to efficiently and effectively eliminate nonpromising uncertain objects. Our filtering techniques are based on an R-Tree.
—We show, by a theoretical analysis and experiments, that the expected sizes of lskyline and gskyline are bounded by that of skyline over certain data. Our experiments also show that the sizes of lskyline and gskyline are similar, though in the worst case, the size of gskyline could be much larger than that of lskyline. These guarantee that stochastic skylines can be used as effectively as conventional skyline for personal trade-offs.

Besides the theoretical results, our extensive experiments on real and synthetic data are conducted to demonstrate the efficiency of our techniques.

The rest of the article is organized as follows. In Section 2, we formally define the problem and present preliminaries including a size estimation of stochastic skylines. In Section 3, we show that the problem of determining the lower orthant order between two objects is NP-complete regarding the dimensionality, and then present an efficient verification algorithm that runs in polynomial time if d is fixed. Section 4 shows the problem of testing the usual order can be conducted in polynomial time and presents a novel paradigm to speed up the testing algorithm. In Section 5, we present our framework, and novel filtering techniques. Experiment results are presented in Section 6. Section 7 provides the related work and Section 8 concludes the work. Note that to enhance the readability, we put some lengthy proofs in the Appendix.

Table I. The Summary of Notations

Notation	Definition
stochastic skylines	stochastic skylines (e.g., lskyline and gskyline)
lskyline	stochastic skyline based on lower orthant order
gskyline	stochastic skyline based on usual order
U, V	uncertain objects in R_+^d
u, v (x, y)	instances (points) of uncertain objects in R_+^d
n	number of uncertain objects
U_{mbb}	the MBB of U
U_{max} (U_{min})	upper (lower) corner of the U_{mbb}
$R(x, y)$	rectangular region with x (y) as lower (upper) corner
R_x	rectangular region with the origin and x as lower and upper corners
$U.cdf$	the cumulative distribution function of U
$\mu_i(U)$	the *mean* of U on the i-th dimension
$\sigma_i^2(U)$	the *variance* of U on the i-th dimension
$U \prec_{lo} V$	U stochastically dominates V regarding lower orthant orders
$U \prec_{uo} V$	U stochastically dominates V regarding usual orders

2. BACKGROUND INFORMATION

Table I summarizes the mathematical notations used throughout this article.

2.1. Problem Definition

We use R_+^d to denote the points in R^d with nonnegative coordinate values. Without loss of generality, in the article we only consider the R_+^d space. A point (instance) x referred in the article, by default, is in R_+^d.

Let the ith coordinate value of x be denoted by x_i. Given two points x and y, x *dominates* y (denoted by $x \prec y$) if $x_i \leq y_i$ for $1 \leq i \leq d$ and there is a $j \in [1, d]$ such that $x_j < y_j$. We use $x \preceq y$ to denote the case that either x equals y on each dimension or x dominates y, and use $R(x, y)$ to denote a rectangular region in R_+^d where x and y are lower and upper corners, respectively.

An uncertain object can be described either *continuously* or *discretely*. As discussed earlier, in this article we focus on discrete cases and objects are on R_+^d. That is, an uncertain object U consists of a set $\{u_1, \ldots, u_m\}$ of instances (points) in R_+^d where for $1 \leq i \leq m$, u_i is in R_+^d and occurs with the probability p_{u_i} ($p_{u_i} > 0$), and $\sum_{i=1}^m p_{u_i} = 1$.

For a point $x \in R_+^d$, the *probability mass* $U.cdf(x)$ of U is the sum of the probabilities of the instances in $R((0, \ldots, 0), x)$ where $(0, 0, \ldots, 0)$ is the origin in R^d; that is, $U.cdf(x) = \sum_{u \preceq x, u \in U} p_u$.

Example 1. Suppose the instances of each object in Figure 1(b) have the appearance probabilities as described in the example depicted in Figure 1(a). Then, $A.cdf(x) = \frac{1}{2}$, $B.cdf(x) = \frac{1}{2}$ and $C.cdf(x) = \frac{1}{100}$.

Given a set $S \subseteq R_+^d$, $U.cdf(S)$ denotes the probability mass of U restricted to S; that is, $U.cdf(S) = \sum_{u \in S \cap U} p_u$.

Example 2. As depicted by Figure 2, there are 3 objects A, B, and C. Regarding the shaded area S_1 in Figure 2(a), $A.cdf(S_1) = \frac{1}{2}$, $B.cdf(S_1) = \frac{1}{2}$, and $C.cdf(S_1) = \frac{1}{4}$, while regarding the shaded area S_2 in Figure 2(b), $A.cdf(S_2) = \frac{1}{7}$, $B.cdf(S_2) = 0$, and $C.cdf(S_2) = \frac{1}{4}$.

The *minimal bounding box* U_{mbb} of U is the minimal rectangular region S such that $U.cdf(S) = 1$. For presentation simplicity, the *lower* (*upper*) corner of U_{mbb} is denoted by U_{min} (U_{max}) where the lower (upper) corner of U_{mbb} is the point in U_{mbb} with the

minimum (maximum) coordinate value on each dimension, and uncertain objects are sometimes abbreviated to "objects".

Lower Orthant Order and lskyline. We first present lower orthant orders [Shaked and Shanthikumar 2007].

Definition 1 (*Lower Orthant Order*). Given two uncertain objects U and V, U *stochastically dominates* V regarding the lower orthant order, denoted by $U \prec_{lo} V$, if $U.cdf(x) \geq V.cdf(x)$ for each point $x \in R_+^d$ and $\exists y \in R_+^d$ such that $U.cdf(y) > V.cdf(y)$.

Definition 2 (*lskyline*). Given a set of uncertain objects $\mathcal{U}$, $U \in \mathcal{U}$ is a lskyline object if there is no object $V \in \mathcal{U}$ such that $V \prec_{lo} U$. The set of all lskyline objects of $\mathcal{U}$ is the lskyline of $\mathcal{U}$.

Regarding the example in Figure 1, it can be immediately verified that $A \prec_{lo} B$, $B \not\prec_{lo} C$, $C \not\prec_{lo} B$, $C \not\prec_{lo} A$, and $A \not\prec_{lo} C$. lskyline consists of A and C. Regarding the example in Figure 2, lskyline consists of only one object A.

Usual Order and gskyline. For the ease of a comparison with the lower orthant order, we define the *usual stochastic order* [Shaked and Shanthikumar 2007] (usual order for short) on lower sets.

Definition 3 (*Lower Set*). A set S of points in R_+^d is a *lower set* if for each pair of points $x \preceq y$ in R_+^d, $y \in S$ implies $x \in S$.

Example 3. The shaded areas in Figure 2(a) and (b) are lower sets, respectively.

Definition 4 (*Usual Order*). Given two uncertain objects U and V, U *stochastically dominates* an object V regarding the usual order, denoted by $U \prec_{uo} V$, if $U.cdf(S) \geq V.cdf(S)$ for each lower set $S \subseteq R_+^d$ and $U.cdf(S') > V.cdf(S')$ for a lower set $S' \subseteq R_+^d$.

Example 4. Regarding the depicted lower set S_1 (shaded area) in Figure 2(a), since $A.cdf(S_1) = \frac{1}{2}$ and $C.cdf(S_1) = \frac{1}{4}$, $C.cdf(S_1) < A.cdf(S_1)$; thus, $C \not\prec_{uo} A$. Regarding the depicted lower set S_2 (shaded area) in Figure 2(b), since $A.cdf(S_2) = \frac{1}{7}$ and $C.cdf(S_2) = \frac{1}{4}$, $A.cdf(S_2) < C.cdf(S_2)$; thus, $A \not\prec_{uo} C$.

Note that the usual order in Shaked and Shanthikumar [2007] is defined over *upper sets*; that is, in Definition 4, lower sets are replaced by upper sets, and $\geq$ and $>$ are replaced by $\leq$ and $<$, respectively. It is well-known (see wikipedia for example) that the complement of any lower set is an upper set, and vice versa. Consequently, it is immediate that Definition 4 is equivalent to the usual order definition in Shaked and Shanthikumar [2007].

Definition 5 (*gskyline*). In a set $\mathcal{U}$ of uncertain objects, $U \in \mathcal{U}$ is a gskyline object if there is no object $V \in \mathcal{U}$ such that $V \prec_{uo} U$. The set of all gskyline objects of $\mathcal{U}$ is the gskyline of $\mathcal{U}$.

Problem Statement. In this article we investigate the problem of efficiently computing lskyline and gskyline of a set of uncertain objects, respectively.

2.2. Preliminary

We say that U stochastically equals V regarding the lower orthant order, denoted by $U =_{lo} V$, if $U.cdf(x) = V.cdf(x)$ for any point $x \in R_+^d$. Similarly, U stochastically equals V regarding the usual order, denoted by $U =_{uo} V$, if $U.cdf(S) = V.cdf(S)$ for any lower set S in R_+^d. Given two objects U and V, we define $U = V$ if there is a one to one mapping h from U to V such that for each instance u of U, $u = h(u)$ and $p_u = p_{h(u)}$; that is, h preserves the locations and probabilities. The following lemma states that $=_{lo}$

($=_{uo}$) is equivalent to $=$ between two uncertain objects; it can be immediately verified based on the definition of $U.cdf$ and $V.cdf$.

LEMMA 1. *Given two (uncertain) objects U and V, $U =_{lo} V$ ($U =_{uo} V$) if and only if $U = V$.*

LEMMA 2. *Given two objects U and V where $U \neq V$, $U \prec_{lo} V$ ($U \prec_{uo} V$) if and only if for any x (lower set S) in R_+^d, $U.cdf(x) \geq V.cdf(x)$ ($U.cdf(S) \geq V.cdf(S)$).*

Lemma 2 states that when $U \neq V$, we can remove the condition " $\exists y \in R_+^d$ such that $U.cdf(y) > V.cdf(y)$" from Definition 1, and "$U.cdf(S') > V.cdf(S')$ for a lower set $S' \subseteq R_+^d$" from Definition 4, respectively.

Minimality of Stochastic Skylines. A function f in R_+^d is decreasing if $\forall x, y \in R_+^d, x \preceq y$ implies $f(x) \geq f(y)$. The following theorems are proved in Shaked and Shanthikumar [2007, pages 309 and 266, respectively].

THEOREM 1. *Let $U = (U_1, \ldots, U_d)$ and $V = (V_1, \ldots, V_d)$ be two d-dimensional independent random vectors to describe objects U and V, respectively, where U_i and V_i are subvariables of U and V on ith-dimension. Assuming $U \neq V$, then $U \prec_{lo} V$ if and only if $E[\prod_{i=1}^d f_i(U_i)] \geq E[\prod_{i=1}^d f_i(V_i)]$ for every collection $\{f_i \mid 1 \leq i \leq d\}$ of univariate nonnegative decreasing functions where expectations exist.*

Theorem 1 implies that the lower orthant order is the stochastic order over the family of all nonnegative multiplicative decreasing functions. Consequently, lskyline provides the superior set of objects in a set $\mathcal{U}$ of uncertain objects, regarding the expected utility principle and all nonnegative multiplicative decreasing functions, with the minimum size.

THEOREM 2. *Let U and V be two d-dimensional independent random vectors to describe objects U and V, respectively. Assuming $U \neq V$, then $U \prec_{uo} V$ if and only if $E[f(U)] \geq E[f(V)]$ for any f in the family of nonnegative decreasing functions where expectations exist.*

Theorem 2 implies that the usual order is the stochastic order over the family of all decreasing functions. Consequently, gskyline provides the superior set of objects in a set $\mathcal{U}$ of uncertain objects, regarding the expected utility principle and all decreasing functions, with the minimum size.

gskyline, lskyline, and pskyline. Recall that for an $x \in R_+^d$, R_x denotes the rectangular region in R_+^d where the origin $(0, 0, \ldots, 0)$ and x are lower and upper corners, respectively (e.g., the shaded region in Figure 2(a)). Clearly, R_x is one type of lower sets. Thus, $U \not\prec_{lo} V$ implies $U \not\prec_{uo} V$. Therefore, lskyline is a subset of gskyline. Note that the two examples in Section 1 show that a threshold-based probabilistic skyline (pskyline) model [Atallah et al. 2011; Pei et al. 2007] is *orthogonal* to gskyline and lskyline models, respectively; that is, while precisely providing the skyline probabilities, pskyline is not modeled to provide the candidature of optimal solutions regarding the expected utility principle. Figure 3 summarizes the relationships among lskyline, gskyline, and a threshold-based pskyline [Pei et al. 2007].

While lskyline provides the minimum candidate set to the optimal solutions of all monotonic (decreasing) multiplicative functions, lskyline may fail to cover the optimal solutions of some monotonic (decreasing) functions in other forms (e.g., linear functions) as shown by the example in Section 1. Table II summarizes all the situations towards whether gskyline (GS), lskyline (LS), and a threshold-based pskyline (PS) may serve as a candidate set to the family of all monotonic functions (general F) and the family of all monotonic multiplicative functions (m.p F), respectively.

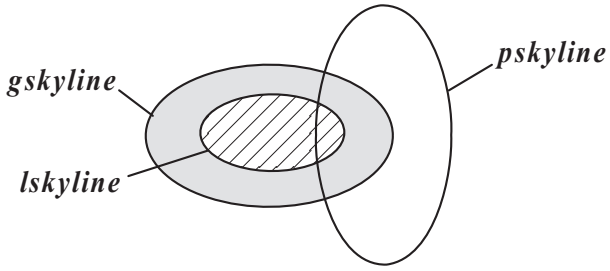

Fig. 3. Relationship among 3 skylines.

Table II. Candidature of Different Skyline Models

	GS	LS	PS
general F	yes & minimum	no	N/A
m.p F	yes	yes & minimum	N/A

Size Estimation. It tends to be quite complicated to estimate the size of lskyline (gskyline) of $\mathcal{U}$ when each object is described by discrete cases. Below, we show that if each object $U \in \mathcal{U}$ follows a continuous distribution with the uniform assumption, the expected number of lskyline (gskyline) objects is bounded by $\ln^d(n)/(d+1)!$, where $\ln^d(n)/(d+1)!$ is the expected size of conventional skyline in $(d+1)$ dimensional space [Chaudhuri et al. 2006], and n is the number of objects in $\mathcal{U}$.

THEOREM 3. *Given a set of objects $\mathcal{U}$, assume that MBBs of all objects are hyper-cubes. We assume that the pdf of an object is continuous with the uniform distribution (i.e., a constant in its MBB), and the lengths of the MBBs and lower corners of the MBBs on each dimension are independent and follow the same distribution. Then the expected sizes of lskyline and gskyline are both bounded by* $\ln^d(n)/(d+1)!$.

PROOF. $\forall U \in \mathcal{U}$, let $l(U)$ denote the length of the hyper-cubes respectively. Clearly, $(U_{min}, l(U))$ is a point in $(d+1)$-dimensional space. Since the pdf of each object follows the uniform distribution, it can be immediately verified that if $(U_{min}, l(U)) \prec (V_{min}, l(V))$, then U stochastically dominates V. Consequently, the number of objects on gskyline of $\mathcal{U}$ is not greater than the size of skyline of $\{(U_{min}, l(U))|U \in \mathcal{U}\}$. Since $\{(U_{min}, l(U))|U \in \mathcal{U}\}$ is a set of points on a $(d+1)$-dimensional space such that every coordinate is independent and follows the same distribution, the expected skyline size of $\{(U_{min}, l(U))|U \in \mathcal{U}\}$ is bounded by $\ln^d(n)/(d+1)!$ [Chaudhuri et al. 2006].

Immediately, the same argument also holds for gskyline. Thus, the expected size of gskyline is also bounded by $\ln^d(n)/(d+1)!$. □

Note that in the worst case, the ratio of the size of gskyline over the size of lskyline could be as bad as n, where n is the number of uncertain objects. Shortly we give such an example by Figure 4; there are four objects and each object has two instances with the same occurrence probability (0.5). According to the definition of lskyline and gskyline, A is the only lskyline object and all objects (A,B,C,D) are gskyline objects. Following the similar rationale, we can come up with the examples with n objects in which A is the only lskyline object and all n objects are not dominated (general order) by any other object if their coordinate values (except A) are in reverse order in dimension x and dimension y.

Nevertheless, the empirical study in Section 6 shows that the size of lskyline and the size of gskyline in a d-dimension space are very close in practice, and are most likely bounded by the size of conventional skyline in a $(d+1)$-dimensional space.

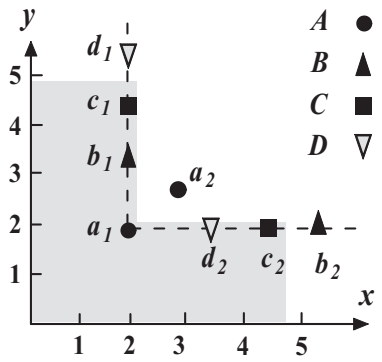

Fig. 4. #gskyline VS #lskyline.

2.3. Framework

Our techniques for computing the lskyline (gskyline) of a set $\mathcal{U}$ of objects follow the standard two-phase framework: pruning and verification. In the pruning phase, efficient and effective techniques are developed to prune nonpromising objects. In the verification phase, we present novel and efficient algorithms to test if $U \prec_{lo} V$ and $U \prec_{uo} V$, respectively, for two uncertain objects U and V. The algorithm to test if $U \prec_{lo} V$ runs in polynomial time regarding the number of instances in U and V when the dimensionality is fixed; and the algorithm to test if $U \prec_{uo} V$ between two uncertain objects runs in polynomial time regarding the dimensionality, and the number of instances in U and V.

3. TESTING LOWER ORTHANT ORDER

According to Lemma 1, testing if $U =_{lo} V$ is equivalent to testing if $U = V$; thus it can be conducted in $O(dm \log m)$ if instances are firstly sorted according to the lexicographic order where m is the number of instances in U (V). Given two objects U and V, in this section we present an efficient algorithm to determine whether U stochastically dominates V if $U \neq V$.

3.1. Testing Finite Number of Points Only

Naively following Definition 1 to compute $U.cdf(x)$ and $V.cdf(x)$ for every point x in R_+^d is computationally infeasible since an infinite number of check points is involved.

A point $x \in R_+^d$ is a *violation point* regarding $U \preceq_{lo} V$ if $U.cdf(x) < V.cdf(x)$. Assuming that $U \neq V$, it immediately follows from Definition 1 and Lemma 2 that U does not stochastically dominate V regarding the lower orthant order if and only if there is a violation point regarding $U \prec_{lo} V$. Consequently, determining if $U \prec_{lo} V$ is converted to determining if there is a violation point regarding $U \prec_{lo} V$, given $U \neq V$.

An intuitive way by checking every instance v in V to find a violation point does not work. As depicted in Figure 5, U consists of two instances u_1 and u_2 with $p_{u_1} = p_{u_2} = \frac{1}{2}$, V consists of 3 instances v_1, v_2, and v_3 with $p_{v_1} = p_{v_2} = p_{v_3} = \frac{1}{3}$, and v_3 is placed at the same position of u_2. It can be immediately verified that for $1 \leq i \leq 3$, $U.cdf(v_i) \geq V.cdf(v_i)$. Nevertheless, we cannot conclude that $U \prec_{lo} V$ since x in Figure 5 is a violation point regarding $U \prec_{lo} V$; that is, $V.cdf(x) > U.cdf(x)$.

For an object V in R_+^d, we use $\mathcal{D}_i(V)$ $(1 \leq i \leq d)$ to denote the set of all distinct ith coordinate values of the instances of V. Then, $\prod_{i=1}^{d} \mathcal{D}_i(V)$ forms a grid; see Figure 6(a) as an example. Theorem 4 next states that we only need to check every (grid) point in $\prod_{i=1}^{d} \mathcal{D}_i(V)$ to determine if there is a violation point regarding $U \prec_{lo} V$.

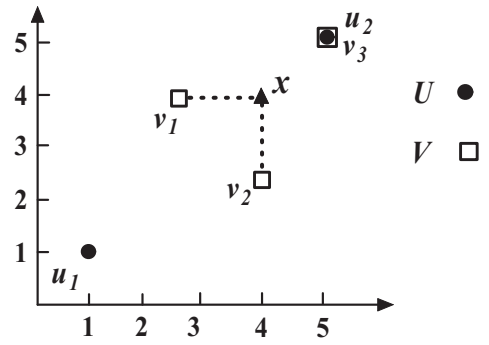

Fig. 5. An example.

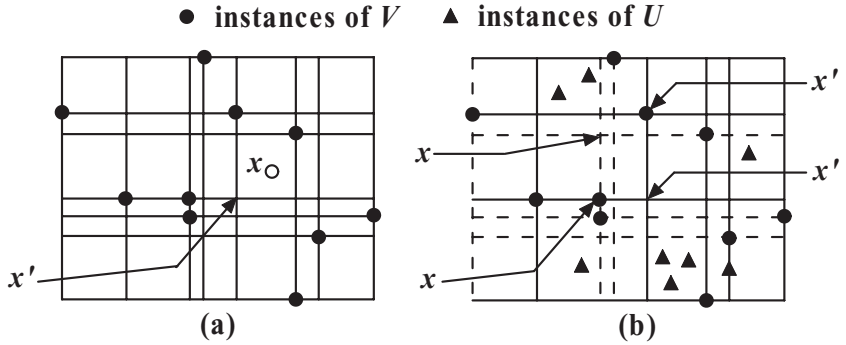

Fig. 6. Grid points.

THEOREM 4. *Suppose that $U \neq V$. Then, V is not stochastically dominated by U regarding the lower orthant orders if and only if there is a (grid) point $x \in \prod_{i=1}^{d} \mathcal{D}_i(V)$ that is a violation point regarding $U \prec_{lo} V$ (i.e., $V.cdf(x) - U.cdf(x) > 0$).*

PROOF. The "if" part is immediate according to Definition 1. Next we prove the "only if" part.

Suppose that V is not stochastically dominated by U regarding the lower orthant orders. Then, based on Definition 1 and Lemma 2 there is a violation point regarding $U \prec_{lo} V$. It can be immediately verified that if there is a violation point regarding $U \prec_{lo} V$ then there must be a violation point x in V_{mbb}. Let x' be the lower corner of the grid cell in $\prod_{i=1}^{d} \mathcal{D}_i(V)$ which contains x; see Figure 6(a) for example. It is immediate that $V.cdf(x') = V.cdf(x)$ and $U.cdf(x') \leq U.cdf(x)$. Since $U.cdf(x) < V.cdf(x)$, $U.cdf(x') < V.cdf(x')$. □

Naive Algorithm. A naive algorithm is to check every grid point to calculate its $U.cdf$ and $V.cdf$; it terminates when we find a violation point or all grid points are exhausted. It is a correct algorithm, if $U \neq V$, based on Theorem 4. Note that there are at most m^d grid points in $\prod_{i=1}^{d} \mathcal{D}_i(V)$ where m is the number of instances in V. Consequently, the naive algorithm needs to check $O(m^d)$ grid points for computing $U.cdf$ and $V.cdf$.

3.2. NP-Completeness

In fact we can show that the exponential time complexity regarding d is unavoidable by Theorem 5 given shortly. The proof of Theorem 5 is lengthy and nontrivial which

will be presented in Appendix A.1. The basic idea is to convert the minimum set cover problem (decision version) [Garey and Johnson 1990] to a special case of testing whether $U \not\prec_{lo} V$.

THEOREM 5. *Given two objects U and V, assume that $U \neq V$. Then, the problem of determining whether $U \not\prec_{lo} V$ is NP-complete regarding the dimensionality d.*

3.3. Efficient Testing Algorithm

The NP-completeness implies that the exponential time complexity regarding d in the worst case cannot be improved. In this section, we present two techniques that may potentially reduce the number of grid points to be tested in practice.

Switching Distinct Values Only. For $1 \leq i \leq d$, let the set of distinct values of V on ith dimension, $\mathcal{D}_i(V) = \{a_1, a_2, \ldots, a_l\}$, be sorted increasingly; that is, $a_i < a_{i+1}$ for $1 \leq i \leq l-1$. a_j $(1 \leq j \leq l)$ is a *switching* distinct value regarding U if there is at least one value in $\mathcal{D}_i(U)$ that is in $[a_j, a_{j+1}]$ when $j < l$ or in $[a_l, \infty)$ when $j = l$. Let $D_i^s(V)$ denote the set of switching distinct values of $\mathcal{D}_i(V)$ regarding U. The following theorem implies that only the (grid) points in $\prod_{i=1}^d D_i^s(V)$, instead of $\prod_{i=1}^d \mathcal{D}_i(V)$, need to be checked. For example, the grid framed by the solid lines in Figure 6(b) depicts $\prod_{i=1}^d D_i^s(V)$. Clearly, $D_i^s(V) \subseteq \mathcal{D}_i(V)$; consequently, we may significantly reduce the number of grid points for testing.

THEOREM 6. *Assume that $U \neq V$. Then, V is not stochastically dominated by U regarding the lower orthant orders if and only if there is a (grid) point $x \in \prod_{i=1}^d D_i^s(V)$ such that $V.cdf(x) - U.cdf(x) > 0$.*

PROOF. Note that $\prod_{i=1}^d D_i^s(V) \subseteq \prod_{i=1}^d \mathcal{D}_i(V)$. It can be immediately verified that for each grid point $x \in \prod_{i=1}^d \mathcal{D}_i(V)$ but not in $\prod_{i=1}^d D_i^s(V)$, there is a grid point x' in $\prod_{i=1}^d D_i^s(V)$ such that $x \prec x'$ and the instances of U falling in $R((0, \ldots, 0), x)$ are the same as those of U falling in $R((0, \ldots, 0), x')$; see such two pairs in Figure 6(b) for an example. Therefore, $U.cdf(x') = U.cdf(x)$ and $V.cdf(x') \geq V.cdf(x)$. Thus, this theorem immediately follows from Theorem 4. □

Discarding a Rectangular Region. Let $R(x, y)$ denote a rectangular region in R_+^d where the lower and upper corners are x and y, respectively. $R(x, y)$ is valid regarding $U \prec_{lo} V$ if each point in $R(x, y)$ is valid regarding $U \prec_{lo} V$. Theorem 7 provides a sufficient condition for $R(x, y)$ to be valid. Consequently, $R(x, y)$ can be discarded from the procedure of finding a violation point.

THEOREM 7. *Given an $R(x, y) \in R_+^d$, suppose that $U.cdf(x) \geq V.cdf(y)$. Then, for every point z in $R(x, y)$ (i.e., $x \preceq z \preceq y$), $U.cdf(z) \geq V.cdf(z)$.*

PROOF. Since $x \preceq z \preceq y$, $U.cdf(z) \geq U.cdf(x)$ and $V.cdf(y) \geq V.cdf(z)$. Thus, the theorem holds. □

To facilitate the observation in Theorem 7, our algorithm is partitioning-based, which iteratively divides rectangular regions into disjoint subrectangular regions. We use a queue Q to maintain a set of disjoint rectangular regions that cannot be discarded by Theorem 7 (i.e., not yet valid regarding $U \prec_{lo} V$). Then, for each rectangular region $R(x, y) \in Q$ the algorithm checks if one of the latest generated corner points of $R(x, y)$ is a violation point regarding $U \prec_{lo} V$. If none of them is a violation point, then the algorithm checks if $R(x, y)$ should be discarded or should be split into two disjoint subrectangular regions to be put into Q. Note that the corner points of a $R(x, y)$ are the points $(z_1, z_2, \ldots, z_d)$ such that for $1 \leq i \leq d$, z_i is x_i or y_i where $x = (x_1, \ldots, x_d)$

ALGORITHM 1: Verification($U \prec_{lo} V$)

```
   Input  : objects U and V
   Output : if U ≺lo V (i.e., true or false)
 1 for i = 1 to d do
 2     D_i^s(V) := getSwitchingDistinct(V, U);
 3 if Initial_Check (U ≺lo V) returns false then
 4     return false
 5 mark each corner point of V_mbb as new; Q := {V_mbb};
 6 while Q ≠ ∅ do
 7     r := Q.deque();
 8     calculate V.cdf(x) and U.cdf(x) for each new corner point x of r;
 9     if a new corner of r is a violation point then
10         return false ;
11     if r cannot be discarded and r is not an atom then
12         Q := Q ∪ Split(r);
13 return true
```

and $y = (y_1, y_2, \ldots, y_d)$. The algorithm terminates if a violation point is found or Q is $\emptyset$. Algorithm 1 shortly presents our partitioning-based testing techniques. The algorithm outputs "true" if $U \prec_{lo} V$ and "false" otherwise.

In Algorithm 1, line 2 is to get the set $D_i^s(V)$ of switching distinct values of V regarding U on each dimension i. Line 3 conducts a quick check on each dimension i as follows. Let $U = (U_1, \ldots, U_d)$ and $V = (V_1, \ldots, V_d)$ where U_i and V_i $(1 \leq i \leq d)$ are ith subvariables of U and V, respectively. For each distinct value a_j in $D_i^s(V)$ $(1 \leq i \leq d)$, Initial_Check $(U \prec_{lo} V)$ calculates the total probability, denoted by $V.cdf(V_i \leq a_j)$, of the instances of V with its ith-coordinate value not greater than a_j; similarly, $U.cdf(U_i \leq a_j)$ is also calculated for U. If $V.cdf(V_i \leq a_j) > U.cdf(U_i \leq a_j)$ regarding the currently encountered a_j, then $U \not\prec_{lo} V$ and Initial_Check $(U \prec_{lo} V)$ returns *false*. The correctness of Initial_Check $(U \prec_{lo} V)$ immediately follows from Definition 1.

Line 8 calculates $V.cdf(x)$ and $U.cdf(x)$ for a new corner x of r. Line 9 checks whether one of the new corners (grid points) is a violation point and then removes their "new" marks afterwards. In line 11, the algorithm checks whether r is valid regarding $U \prec_{lo} V$ (i.e., the condition in Theorem 7) and thus can be discarded. r is an *atom* if it cannot be further split to generate new grid points; that is, on each dimension i, r contains at most 2 values from $D_i^s(V)$.

Split(r) in line 12 splits r into two subregions as follows. Note that for each dimension i $(1 \leq i \leq d)$, the values in $D_i^s(V)$ contained by r must be consecutive and are denoted by $r \cap D_i^s(V)$. Firstly, a dimension i is chosen such that $|r \cap D_i^s(V)|$ is maximized. Then, the two median values $l_1 < l_2$ in $r \cap D_i^s(V)$ are chosen to split r into r_1 and r_2; that is, the points in r with the ith coordinate value not greater than l_1 belong to r_1, and the others in r belong to r_2. The two median values are used for splitting since we want to "maximize" the number of grid points to be discarded in case if one of r_1 and r_2 is valid regarding $U \prec_{lo} V$. Moreover, Split(r) also marks the newly generated corners of r_1 and r_2; that is, the corners of r_1 and r_2 are not the corners of r. Note that there is an extreme case; while splitting on the ith dimension, if r contains only 3 values in $D_i^s(V)$, then splitting r into r_1 and r_2 on the ith dimension leads to that one of r_1 and r_2 degenerates into a rectangular region in a $(d-1)$ space.

Example 5. As depicted in Figure 7(a), assume that V has 5 switching distinct values in each dimension. Algorithm 1 splits V_{mbb} into r_1 and r_2 in Figure 7(b). Suppose

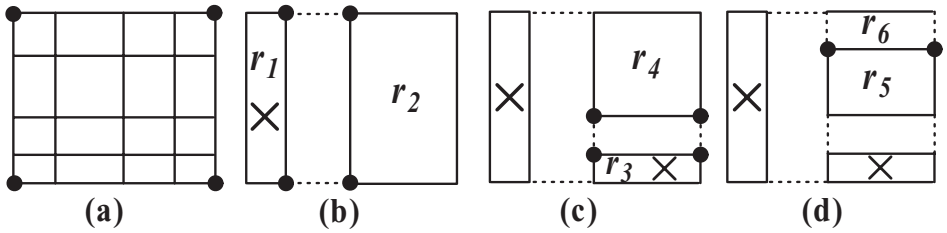

Fig. 7. An example for splitting.

that r_1 is discarded by line 11. Algorithm 1 further splits r_2 into r_3 and r_4 in Figure 7(c). Again, assume that r_3 is discarded. r_4 is split into r_5 and r_6 by the algorithm where r_6 is a line segment containing 3 grid points. Any further splitting on r_6 will be conducted on the line segment. In this example, at each iteration, solid points in Figure 7 indicate the newly generated grid points.

Based on the correctness of Initial_Check ($U \prec_{lo} V$), Theorems 4, 6, and 7, Algorithm 1 is correct when $U \neq V$.

Time Complexity. Algorithm 1 intensively involves the computation of cumulative probabilities of U and V over a rectangular region. To speed up such computation, the instances of each object U are organized by an in-memory aggregate R-tree, called *local aggregate R-tree* of U, where each entry records the sum of the probabilities of instances contained. Then, the window aggregate techniques in Tao and Papadias [2012] are employed in our algorithm to calculate various $U.cdf$ and $V.cdf$.

getSwitchingDistinct(V, U) is conducted as follows. Note that $\mathcal{D}_i(V)$ is sorted increasingly ($1 \leq i \leq d$). Let $\mathcal{D}_i(V) = \{a_1, \ldots, a_l\}$. Since the occurrence probability of each instance is positive, based on the definition of switching distinct values it is immediate that a_j ($j < l$) is a switching distinct value if and only if $U.cdf(U_i < a_j) < U.cdf(U_i \leq a_{j+1})$; here, we calculate $U.cdf(U_i \leq a_j)$ and $U.cdf(U_i < a_j)$ by the window aggregate techniques in Tao and Papadias [2012]. We also calculate $V.cdf(V_i \leq a_j)$ for each $a_j \in D_i^s(V)$ to conduct Initial_Check ($U \prec_{lo} V$). Clearly, lines 1–4 totally run in time $O(dm \log m + dm(T(U_{artree}) + T(V_{artree})))$ where $T(U_{artree})$ and $T(V_{artree})$ are the costs to conduct window aggregates over the local aggregate R-trees of U and V, respectively, m is the number of instances in V, and $O(m \log m)$ is the time complexity to get each $\mathcal{D}_i(V)$.

Assuming that $D_i^s(V)$ is stored in an array. Since it is sorted, it takes constant time to find the two splitting values in Split(r) along a dimension. Since Algorithm 1 calculates $U.cdf(x)$ and $V.cdf(x)$ only once at each grid point x in $\prod_{i=1}^{d} D_i^s(V)$, the total time spent in calculating $U.cdf$ and $V.cdf$ at all grid points is $O((T(U_{artree}) + T(V_{artree})) \prod_{i=1}^{d} k_i)$ where $k_i = |D_i^s(V)|$. Clearly, checking if r is valid regarding $U \prec_{lo} V$ is only invoked when one new grid point (corner) is generated; consequently, the total time for such a check is $O(\prod_{i=1}^{d} k_i)$. Thus, the time complexity from line 5 to the end is $O((T(U_{artree}) + T(V_{artree})) \prod_{i=1}^{d} k_i)$. The following theorem is immediate based on the preceding discussions.

THEOREM 8. *Algorithm 1 runs in time* $O(dm \log m + m^d(T(U_{artree}) + T(V_{artree})))$ *where* m *is the number of instances in* V.

The naive algorithm in Section 4.1 can also employ the window aggregate techniques in Tao and Papadias [2012] to calculate $U.cdf$ and $V.cdf$. Although the time complexity of Algorithm 1 is similar to that of the naive algorithm in the worst case, our experiment

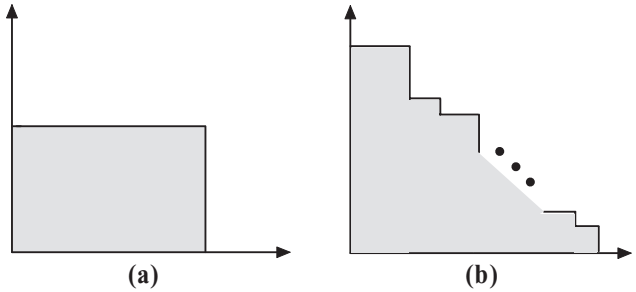

Fig. 8. Lower sets.

in Section 6 demonstrates that the naive algorithm is impractical, while Algorithm 1 is very efficient in practice.

4. TESTING USUAL ORDER

In this section, we investigate the problem of efficiently testing if $U \prec_{uo} V$. Based on the definitions of the lower orthant order and the usual order, when $U \prec_{lo} V$, it could be either $U \prec_{uo} V$ or $U \not\prec_{uo} V$ as shown by the example in Figure 2. Therefore, the testing techniques for the lower orthant order are not applicable to testing the usual order.

In fact, the usual order involves more complicated geometric terms than the lower orthant order as discussed in Section 1. It can be immediately verified that any combination of the rectangles with the origin as the lower corner is a lower set and vice versa. As shown in Figure 8, a lower set can be as simple as one rectangle in Figure 8(a) and as complex as the combination of an arbitrary (possibly infinite) number of rectangles in Figure 8 (b), while the lower orthant order always involves a simple lower set: one rectangle (e.g. the one in Figure 8(a)).

The techniques developed in the last section are based on finding a violation point; that is, finding one rectangle with the origin as the lower corner and the violation point as the upper corner that violates Definition 1. Nevertheless, trivially finding a lower set that violates Definition 4 is computationally infeasible since even one lower set could involve a combination of infinite number of rectangles as shown in Figure 8(b). Shortly we show for the first time regarding the existing literature that testing if $U \prec_{uo} V$ can be conducted in polynomial time, regarding the dimensionality and the number of instances in U and V, in contrast to the complexity of testing $U \prec_{lo} V$ which is shown NP-complete regarding the dimensionality. Without loss of generality, we assume $U \neq V$ since testing $U = V$ can be conducted in $O(dm \log m)$ time by firstly sorting instances according to the lexicographic order where m is the number of instances in U and V.

4.1. Finite Computation

Naively following Definition 4 to determine if $U \prec_{uo} V$ involves enumerating an infinite number of lower sets and each lower set may involve a combination of infinite number of rectangles. Theorem 9 next shows that only a finite number of rectangles with the origin as the lower corner should be involved.

THEOREM 9. *Suppose that $U \neq V$. $U \prec_{uo} V$ if and only if for each subset X of V, $U.cdf(\cup_{x \in X} R_x) \geq V.cdf(\cup_{x \in X} R_x)$.*

PROOF. Recall that R_x is the rectangular region with the origin and x as lower and upper corners, respectively. Since $\cup_{x \in X} R_x$ is a lower set, the "only if" part immediately follows from Definition 4.

Suppose that S is a lower set. Let $L_S = \cup_{x \in V \cap S} R_x$. Immediately, $L_S \subseteq S$ and $V.cdf(L_S) = V.cdf(S)$. Since $L_S \subseteq S$, $U.cdf(S) \geq U.cdf(L_S)$. Since $V \cap S \subseteq V$, $U.cdf(L_S) \geq V.cdf(L_S)$ based on the "if" condition in this theorem. Thus, $U.cdf(S) \geq V.cdf(S)$. Hence, the "if" part holds. □

Although Theorem 9 states that the testing may be conducted in finite time, enumerating all subsets of V requires exponential time.

4.2. Reduction to Max-Flow

We convert the problem of testing if $U \prec_{uo} V$ to the *max-flow problem*. The max-flow problem is to find a feasible flow through a single-source, single-sink *network* that is maximized. It is formally defined as follows. Let $G = (N, A)$ be a network with $s, t \in N$ being the *source* and *sink*, respectively, where N is the set of nodes and A is the set of arcs. The *capacity* of an arc $\langle u, v \rangle$ is a nonnegative value $w(u, v)$. A *feasible flow* f of G maps each arc in A to a nonnegative value with the following constraints:

—*Capacity Constraint:* $\forall \langle u, v \rangle \in A$, $f(u, v) \leq w(u, v)$.
—*Flow conservation:* $\forall u \in N - \{s, t\}$, $\sum_{\langle v,u \rangle \in A} f(v, u) = \sum_{\langle u,v \rangle \in A} f(u, v)$.

The value of a feasible flow f is $|f| = \sum_{\langle s,v \rangle \in A} f(s, v)$. The *max-flow* of G is a feasible flow f such that $|f|$ is maximized. It is well-known that the max-flow can be computed in polynomial time (e.g., Hochbaum [2008]).

Reduction to Max-Flow. Next we convert testing if $U \prec_{uo} V$ to the problem of determining if an auxiliary network $G_{U,V}$ has a max-flow f with $|f| = 1$. We construct the auxiliary network $G_{U,V} = (N, A)$ from U and V as follows.

—Each instance $u \in U$ corresponds to the two nodes u^1 and u^2, and an arc $\langle u^1, u^2 \rangle$ with the capacity $w(u^1, u^2) = p_u$ in $G_{U,V}$.
—Each instance $v \in V$ corresponds to the two nodes v^1 and v^2, and an arc $\langle v^1, v^2 \rangle$ with the capacity $w(v^1, v^2) = p_v$ in $G_{U,V}$.
—For each pair of vertices $u \in U$ and $v \in V$, if $u \preceq v$, there is an arc $\langle v^2, u^1 \rangle$ in $G_{U,V}$ with the capacity $w(v^2, u^1) = 2$.
—In $G_{U,V}$, we make the source s connect to each v^1 by an arc $\langle s, v^1 \rangle$ with the capacity $w(s, v^1) = 2$, and make each u^2 connect to the sink t by an arc $\langle u^2, t \rangle$ with the capacity $w(u^2, t) = 2$.

Example 6. The two objects U and V are shown in Figure 9(a) where the 4 instances (u_1, u_2, u_3, and u_4) of U have the same probability $\frac{1}{4}$, and the 3 instances (v_1, v_2, and v_3) of V have the same probability $\frac{1}{3}$. Figure 9(b) shows the corresponding auxiliary network $G_{U,V}$ together with the arc capacities labeled; note that the arcs without capacity labeled in Figure 9(b) have the capacity 2.

The Theorem 10 shortly shows that determining if $U \prec_{uo} V$ is equivalent to testing if the auxiliary network $G_{U,V}$ has a max-flow with the flow value 1. The detailed proof of Theorem 10 will be presented in Appendix A.2.

THEOREM 10. *Suppose that $U \neq V$. $U \prec_{uo} V$ if and only if $G_{U,V}$ has a max-flow f with $|f| = 1$.*

Note that in the reduction to the max-flow problem, it is not necessary to split an instance into two vertices. We present the reduction based on splitting instances for

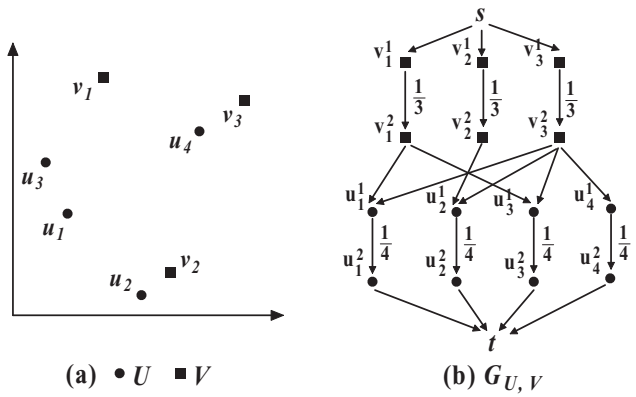

Fig. 9. Reduction to max-flow.

ALGORITHM 2: BasicV ($U \prec_{uo} V$)

Input : objects U and V
Output : if $U \prec_{uo} V$ (i.e., *true* or $false$)
. **if** $U_{min} \preceq V_{min}$ & $U_{max} \preceq V_{max}$ **then**
. *Step 1:* Construct $G_{U,V}$.
. *Step 2:* Run max-flow algorithm [Hochbaum 2008] on $G_{U,V}$ to generate max-flow f. If $|f| = 1$ then return *true*; otherwise, return $false$.
. **else** return $false$.

the ease of presenting our speedup technique in Section 4.4 and also because it does not cause much extra costs according to the existing techniques to determine the max-flow.

4.3. Basic Verification Algorithm

Recall that U_{min} and U_{max} are lower and upper corners of U_{mbb}; and R_x is the rectangular region with the origin as the lower corner and x as the upper corner. We have the following theorem.

THEOREM 11. *Suppose that $U_{min} \not\preceq V_{min}$ or $U_{max} \not\preceq V_{max}$. Then $U \not\prec_{uo} V$.*

PROOF. Note that $U \neq V$. If $U_{max} \not\preceq V_{max}$, then $\exists u \in U$ such that $u \notin R_{V_{max}}$; consequently, $U.cdf(R_{V_{max}}) < 1$. As $V.cdf(R_{V_{max}}) = 1$, $U.cdf(R_{V_{max}}) < V.cdf(R_{V_{max}})$. Thus, $U \not\prec_{uo} V$ as $R_{V_{max}}$ is a lower set.

If $U_{min} \not\preceq V_{min}$, then $\exists v \in V$ such that $U_{min} \not\preceq v$. Thus, $R_v \cap U = \emptyset$; that is, $U.cdf(R_v) = 0$. Since $V.cdf(R_v) \geq p_v > 0$, $U.cdf(R_v) < V.cdf(R_v)$; this means $U \not\prec_{uo} V$. □

Based on Theorems 10 and 11, a basic version of our verification algorithm, Algorithm 2, is outlined next. It outputs "true" if $U \prec_{uo} V$ and "false" otherwise.

The open source-code of the state-of-art max-flow algorithm in Hochbaum [2008] is used in step 2 to compute the max-flow f of $G_{U,V}$ in Algorithm 2. It runs in $O(nm \log m)$ where n is the number of nodes and m is the number of arcs in $G_{U,V}$. Therefore, Algorithm 2 runs in time $O(t_G + nm \log n)$ where t_G is the time to construct $G_{U,V}$. Note that the costs of determining the $\preceq$ relationship for every pair of instances in $U \times V$ are dominant in constructing $G_{U,V}$. Shortly, we present a popular R-tree-based level-by-level paradigm [Brinkhoff et al. 1993] to determine the $\preceq$ relationship for every pair of instances in $U \times V$ instead of naively enumerating every pair in $U \times V$.

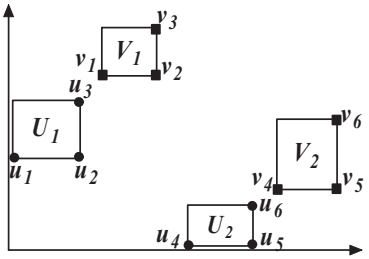

Fig. 10. Level-by-level.

ALGORITHM 3: Determining (FQ)

```
   Input  : T_U (R-tree of U), T_V (R-tree of V)
   Output : FQ
1  (U_mbb, V_mbb) → PQ;
2  while PQ ≠ ∅ do
3  |  dequeue (PQ) → (E_U, E_V) ;
4  |  for each child E of E_U do
5  |  |  for child entry E' of E_V do
6  |  |  |  if E fully dominates E' then
7  |  |  |  |  (E, E') → FQ ;
8  |  |  |  |  else
9  |  |  |  |  |  if E partially dominates E' then (E, E') → PQ ;
```

Level-by-Level. In our algorithm, the instances in every object U are organized by an in-memory R-tree T_U. As depicted in Figure 10, T_U has two intermediate entries U_1 and U_2, and T_V has two intermediate entries V_1 and V_2. As none of the instances in U_2 dominates any instances in V_1, the enumeration of pairs in $U_2 \times V_1$ can be avoided; the level-by-level paradigm can achieve this.

The notation that follows is needed. Given two entries in R-trees (i.e., rectangular regions) E and E', E fully dominates E' if $E_{max} \preceq E'_{min}$; E partially dominates E' if $E_{min} \preceq E'_{max}$ but $E_{max} \not\preceq E'_{min}$; E does not dominate E' if $E_{min} \not\preceq E'_{max}$. Here, E_{min} (E_{max}) is the lower (upper) corner of E. Note that if both E and E' degrade to points, then there is no partial domination, and that E fully dominates E' is equivalent to the $\preceq$ relationship between two points as defined in Section 2.

Example 7. Regarding Figure 10, U_1 fully dominates V_1, and partially dominates V_2; U_2 does not dominate V_1.

The level-by-level computation is outlined next in Algorithm 3. At each level, we maintain a queue PQ to store pairs of entries (E_U, E_V) such that E_U ($\in T_U$) partially dominates E_V ($\in T_V$). We also maintain a queue FQ to store pairs of entries (E_U, E_V) such that E_U ($\in T_U$) fully dominates E_V ($\in T_V$). Algorithm 3 dequeues every pair in PQ for a further exploration. Once PQ is exhausted, FQ stores all the information of $u \preceq v$ in $U \times V$ to be used to construct $G_{U,V}$.

Example 8. Regarding U and V in Figure 10, PQ is initialized into $\{(U_{mbb}, V_{mbb})\}$ by Algorithm 3. Then, line 3 dequeues PQ to get (U_{mbb}, V_{mbb}). PQ becomes $\{(U_1, V_2)\}$ and FQ becomes $\{(U_1, V_1), (U_2, V_2)\}$ after the first iteration (executing line 4 to line 9).

In the next iteration, line 3 dequeues PQ to get (U_1, V_2). Then the execution of line 4 to line 9 gives $PQ = \emptyset$ and updates FQ to $\{(U_1, V_1), (U_2, V_2), (u_1, v_6), (u_2, v_6)\}$. Algorithm 3 thus terminates since $PQ = \emptyset$.

Note that in Algorithm 3, a child entry of a leaf is the leaf itself. That is, if one of T_U and T_V first reaches the leaf level then it stays at the leaf level while the other continues to go down. The Pruning Rule 1 in Section 5.2 and Algorithm 2 ensure that U_{mbb} partially dominates V_{mbb} if U and V enter into step 1.

Theorem 12 can be immediately verified; it states that FQ generated by Algorithm 3 exactly captures all $u \preceq v$ relationships. Consequently, $G_{U,V}$ can be constructed based on the $\preceq$ relationships captured by FQ. While Algorithm 3 runs in time $O(d|U||V|)$, it is well-known that it can avoid the complexity $\Omega(d|U||V|)$ in many cases in practice.

THEOREM 12. *$\forall u \in U$ and $\forall v \in V$, $u \preceq v$ if and only if there is a pair $(E_U, E_V) \in FQ$ such that $u \in E_U$ and $v \in E_V$.*

4.4. Simplifying Network for Quick Testing

Having a large number of arcs in an auxiliary network of U and V not only increases the construction costs but also increases costs of computing the max-flow. While Algorithm 3 can avoid enumerating every pair of instances in $E_U \times E_V$ when E_U does not dominate E_V, it still has to enumerate every pair of instances in $E_U \times E_V$ for each $(E_U, E_V) \in FQ$. Regarding Example 8, FQ leads to total 20 such pairs, including 9 from $U_1 \times V_1$ and 9 by $U_2 \times V_2$. The 20 pairs are converted to 20 arcs in $G_{U,V}$. This is unavoidable due to the requirement of constructing $G_{U,V}$.

Shortly we propose a simplified auxiliary network $G^s_{U,V}$ by treating E_U as a unit and E_V as another unit for each $(E_U, E_V) \in FQ$ while constructing $G^s_{U,V}$. Then we prove $U \prec_{uo} V$ if and only if $G^s_{U,V}$ has a max-flow f with $|f| = 1$. Regarding Example 8, FQ will lead to only 4 arcs in $G^s_{U,V}$, one by (U_1, V_1), one by (U_2, V_2), one by (u_1, v_6), and one by (u_2, v_6).

To construct $G^s_{U,V}$, we treat each in-memory R tree T_U as a network G_U as follows.

—Each entry $E \in T_U$, from the root to leafs, corresponds to the two nodes E^1 and E^2, and an arc $\langle E^1, E^2 \rangle$ with the capacity $w(E^1, E^2) = p_E$ where $p_E = \sum_{u \in E} p_u$.
—Each link from a parent entry E_1 to a child entry E_2 corresponds to an arc $\langle E_1^2, E_2^1 \rangle$ in G_U with the capacity $w(E_1^2, E_2^1) = 2$.

Example 9. Regarding the example in Figure 10, assume that u_1, u_2, and u_3 have the same probability $\frac{1}{5}$; u_4, u_5, u_6 have the same probability $\frac{2}{15}$; v_i (for $1 \le i \le 6$) has the probability $\frac{1}{6}$. G_V and G_U are depicted and framed by boxes in Figure 11(a) and Figure 11(b) where U and V denote the roots of T_U and T_V, respectively. Note that the arcs without capacities labeled have the capacity 2.

Constructing $G^s_{U,V}$. As follows, $G^s_{U,V}$ is constructed by connecting G_V to G_U with the arcs corresponding to pairs of entries in FQ.

—*Connecting the source:* Add an arc $\langle s, V^1 \rangle$ to link the source s to the root V of T_V with the capacity $w(s, V^1) = 2$.
—*Connecting the sink:* Add an arc $\langle u^2, t \rangle$ to link each leaf u in T_U (i.e., an instance in U) to the sink t with the capacity $w(u^2, t) = 2$.
—*Connecting G_V to G_U:* Run Algorithm 3 to get FQ. For each pair (E_U, E_V) of entries in FQ, add an arc $\langle E_V^2, E_U^1 \rangle$ to $G^s_{U,V}$ with the capacity $w(E_V^2, E_U^1) = 2$.

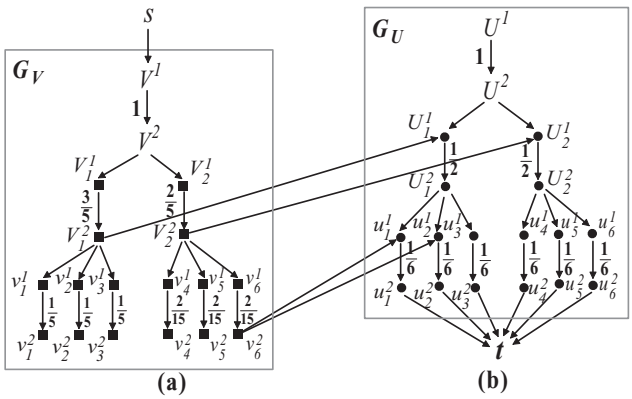

Fig. 11. $G^s_{U,V}$.

Example 10. Continue Example 8 where the generated FQ is $\{(U_1, V_1), (U_2, V_2), (u_1, v_6), (u_2, v_6)\}$. Consequently, the 4 arcs $\langle V_1^2, U_1^1\rangle$, $\langle V_2^2, U_2^1\rangle$, $\langle v_6^2, u_1^1\rangle$, $\langle v_6^2, u_2^1\rangle$ are added to link G_V and G_U as described earlier. Figure 11 shows the result: $G^s_{U,V}$.

Theorem 13 next states that testing if $U \preceq_{uo} V$, when $U \neq V$, is equivalent to determining if $G^s_{U,V}$ has a maximum flow f with $|f| = 1$. The proof of Theorem 13 is presented in Appendix A.3.

THEOREM 13. *Suppose that $U \neq V$. $U \prec_{uo} V$ if and only if $G^s_{U,V}$ has a max-flow f with $|f| = 1$.*

QuickTest. ($U \prec_{uo} V$) is a modification of the basic verification algorithm, Algorithm 2, where $G^s_{U,V}$ is constructed in step 1 and used in step 2 instead of $G_{U,V}$. The correctness of QuickTest immediately follows from Theorem 13.

At each intermediate entry E of an R-tree T_U, we also keep p_E to speed up the construction of $G^s_{U,V}$. While the worst-case time complexity of QuickTest remains the same as that of Algorithm 2, our experiment demonstrates QuickTest can achieve a speedup up to one order of magnitude against Algorithm 2. Note that in constructing $G^s_{U,V}$, we could reverse one of G_U and G_V or both; this gives 3 other ways to construct $G^s_{U,V}$. Nevertheless, our initial experiment shows that they have a similar performance. We do not remove the non-s source nodes and non-t sink nodes from $G^s_{U,V}$ since any nontrivial max-flow algorithm can efficiently handle this.

5. STOCHASTIC SKYLINES COMPUTATION

In this section, we present index-based efficient algorithms to compute two kinds of stochastic skylines, lskyline and gskyline, respectively. They adopt the same framework and same index. We first present the index to be used, followed by our index-based framework, filtering techniques, and algorithm analysis.

5.1. Statistic *R*-Tree

As discussed in Sections 3.3 and 4.4, the instances of an object are organized into a local aggregate R-tree. In our algorithm, we assume that a global R-tree is built on the MBBs of each object; that is, the data entries (unit data) in the global R-tree are MBBs. To facilitate our filtering techniques, we store the following statistic information at each entry of the global R-tree.

Suppose that U has m instances in R_+^d, $u_1, u_2, \ldots, u_m$ with the occurrence probabilities $p_1, p_2, \ldots, p_m$, respectively.

Definition 6 (*Mean* μ). The *mean* of U, denoted by $\mu(U)$, is $\sum_{i=1}^{m} p_i \times u_i$.

Note that $\mu(U)$ is in R_+^d. For $1 \leq i \leq d$, $\mu_i(U)$ denotes the ith coordinate value of $\mu(U)$.

Definition 7 (*Variance* σ^2). For $1 \leq i \leq d$, $\sigma_i^2(U) = \sum_{j=1}^{m} p_j(u_{j,i} - \mu_i(U))^2$ where each $u_{j,i}$ denotes the ith coordinate value of u_j.

Suppose that an entry E of the global R-tree has l child entries $\{E_1, E_2, \ldots, E_l\}$. E stores the MBB of each child entry E_j $(1 \leq j \leq l)$, as well as $\mu_i(E_j)$ and $\sigma_i^2(E_j)$ for $1 \leq i \leq d$. Here, for $1 \leq i \leq d$, $\mu_i(E_j) = \min\{\mu_i(V) \mid V \in E_j\}$ and $\sigma_i^2(E_j) = \max\{\sigma_i^2(V) \mid V \in E_j\}$ are called the *mean* and the *variance* of E_j on ith dimension, respectively.

The global R-tree, together with the aforesaid statistic information, is called a statistic R-tree, denoted by sR-tree. Our algorithmic framework for computing stochastic skylines is conducted against sR-tree of $\mathcal{U}$. To correctly use the verification algorithms (Algorithm 1 and QuickTest in Section 4.4, respectively) in this article we assume that no two objects U and V in an sR tree of $\mathcal{U}$ are equal. In case that $\mathcal{U}$ contains equal objects, we only index one of the equal objects and record the object IDs for others while building an sR-tree of $\mathcal{U}$.

5.2. Framework for Stochastic Skylines Computation

It is immediate that for any lower set $S \in R_+^d$, if $U.cdf(S) \leq V.cdf(S)$ and $V.cdf(S) \leq W.cdf(S)$ then $U.cdf(S) \leq W.cdf(S)$. Immediately, $\prec_{uo}$ has the transitivity. Since $\prec_{lo}$ is defined on a specific type of lower sets, any rectangle with the origin as the lower corner, $\prec_{lo}$ also has the transitivity. Consequently, the standard filtering paradigm is applicable to computing both lskyline and gskyline; that is, if $U \prec_{lo} V$ (or $U \prec_{uo} V$) then V can be immediately removed since for any W, if $V \prec_{lo} W$ (or $V \prec_{uo} W$) then $U \prec_{lo} W$ (or $V \prec_{uo} W$) and W can be pruned by U. Theorem 14 next shows that both stochastic orders also "preserve" the distance.

THEOREM 14. *For two objects U and V, if $dist(U_{min}) < dist(V_{min})$ then $V \not\prec_{lo} U$ and $V \not\prec_{uo} U$, where $dist(U_{min})$ and $dist(V_{min})$ denote the distances of U_{min} and V_{min} to the origin, respectively.*

PROOF. Immediately, $V_{min} \not\prec U_{min}$ and $V_{min} \neq U_{min}$. Thus, there must be an instance u in U such that $V_{min} \not\prec u$ and $V_{min} \neq u$. Therefore, $U.cdf(R_u) > 0$ and $V.cdf(R_u) = 0$ where R_u is the rectangle with the original and u as the lower and upper corners, respectively. Note that R_u is also a lower set. Thus the theorem holds. □

Our index-based algorithm, Algorithm 4, adopts the branch-and-bound search paradigm [Papadias et al. 2003]. While Algorithm 4 that follows is presented against computing lskyline, computing gskyline follows exactly the same procedures by replacing $\prec_{lo}$ with $\prec_{uo}$, and storing gskyline in R_{ssky}. Algorithm 4 iteratively traverses on the global sR-tree to find the data entry (MBB) such that its lower corner has the minimum distance to the origin. An advantage by doing this is that we can guarantee that later accessed objects are with distances to the origin not smaller than those of the earlier accessed objects. Consequently, based on Theorem 14 a later accessed object is only possible to stochastically dominate an earlier accessed object when such distances from two objects are the same. Thus, our algorithm has a progressive nature if all such distances are different. Moreover, when $dist(U_{min}) \neq dist(V_{min})$, our algorithm only needs to test one of "if U stochastically dominates V" and "if V stochastically dominates U" but not both.

ALGORITHM 4: lskyline Computation(sR)

```
   Input  : sR (sR-Tree for 𝒰)
   Output : R_sskyline (lskyline of 𝒰)
 1 R_sskyline := ∅;
 2 QUEUE(the root entry of sR) into a heap H;
 3 while H ≠ ∅ do
 4     E := H.deheap();
 5     if NOT PRUNE (R_sskyline, E) then
 6         if E is an MBB of a V (i.e a data entry) then
 7             for each U ∈ R_sskyline do
 8                 if Verification(U ≺_lo V) then
 9                     Goto Line 3;
10                 else
11                     if dist(U_min) = dist(V_min) then
12                         if Verification(V ≺_lo U) then
13                             R_sskyline := R_sskyline − {U} ;
14             R_sskyline := R_sskyline + {V};
15         else
16             QUEQUE(E) into H;
17 return R_sskyline
```

Lines 2 and 16 push each child entry descriptions of the root or E, including its MBBs and the preceding statistic information, into the heap H. Here, H is a min-heap built against the distances of the lower corners of the MBBs of entries to the origin. PRUNE($R_{sskyline}$, E) returns true if E is pruned by the current $R_{sskyline}$ using our filtering techniques in Section 5.3 for lskyline or the filtering techniques in Section 5.5 for gskyline.

Line 8 (line 12) performs the verification algorithms, Algorithm 1 in Section 3.3 for testing the lower orthant order in computing lskyline or the QuickTest algorithm in Section 4.4 for testing the usual order in computing gskyline. Since U is accessed earlier than V, U is impossible to be stochastically dominated by V unless $dist(U_{min}) = dist(V_{min})$ according to Theorem 14. When $dist(U_{min}) = dist(V_{min})$, according to our algorithm an object U involved in line 12 is not stochastically dominated by any objects accessed earlier regarding the lower orthant order and the usual order, respectively, including those in the current $R_{sskyline}$. Nevertheless, it is possible that V stochastically dominates an earlier accessed object U with $dist(U_{min}) = dist(V_{min})$; such U will be detected and removed from $R_{sskyline}$ by line 12 and line 13.

5.3. Filtering in Computing lskyline

A key in Algorithm 4 is to efficiently and effectively conduct PRUNE($R_{sskyline}$, E). The following two filtering techniques are developed to check if E can be pruned by U for each object U in $R_{sskyline}$ until $R_{sskyline}$ is exhausted or E is pruned.

(1) *MBB-Based Pruning.* The following pruning rule is immediate according to Definition 1 since each object contains at least 2 instances. Note that E_{min} denotes the lower corner of the MBB of an entry E.

PRUNING RULE 1. *If $U_{max} \prec E_{min}$ or $U_{max} = E_{min}$ (i.e., $U_{max} \preceq E_{min}$), then U stochastically dominates every object in E regarding the lower orthant order; that is, E can be pruned.*

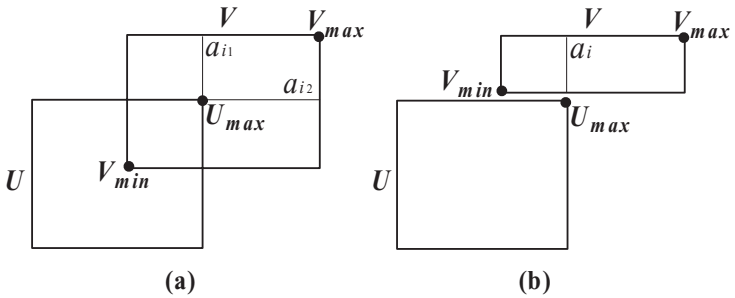

Fig. 12. Statistic pruning for lskyline.

(2) *Statistic-Based Pruning.* Our statistic-based pruning technique uses the observation in Theorem 7 combined with the Cantelli's Inequality [Meester 2004].

Suppose that E cannot be pruned by U by Pruning Rule 1; that is, $U_{max} \not\prec E_{min}$ and $U_{max} \neq E_{min}$. Intuitively, E could still be pruned if U and E are "significantly" separated from the statistic point of view; that is, U_{max} is significantly closer to E_{min} than the mean of E. We show our basic ideas by the case when E is a data entry; that is, E is V_{mbb} - the MBB of an object V.

Suppose that $U_{min} \preceq V_{min}$, $U_{max} \prec V_{max}$, and $U_{max} \not\preceq V_{min}$. Let $U_{max} = (a_1, a_2, \ldots, a_d)$, $V_{min} = (b_1, b_2, \ldots, b_d)$, and I be the subset of $\{1, \ldots, d\}$ such that $a_i > b_i$ if $i \in I$ and $a_i \leq b_i$ otherwise. Note that if $I = \emptyset$, then V is pruned by Pruning Rule 1. Figure 12 shows two cases in a 2-dimensional space where $|I| = 2$ in Figure 12(a) and $|I| = 1$ in Figure 12(b).

Let $k = |I|$. We use the following $k + 1$ rectangular regions to cover V_{mbb}: $\forall i \in I$, $r_i = \{(x_1, \ldots, x_d) \mid x_i \leq a_i, (x_1, \ldots, x_d) \in V_{mbb}\}$, and the $(k + 1)$th rectangular region is $R(U_{max}, V_{max})$. Note that for the ease of statistic estimation that follows, r_i and r_j $(i, j \in I)$ share a common area. It can be immediately verified that $R(U_{max}, V_{max})$ is always a valid region regarding $U \prec_{lo} V$. For each r_i $(i \in I)$, let the lower corner and upper corner of r_i be denoted by $r_{i,min}$ and $r_{i,max}$, respectively. Clearly, $\forall i \in I$, $V.cdf(r_{i,max}) = V.cdf(V_i \leq a_i)$ where V_i is the ith subvariable of V (i.e., $V = (V_1, \ldots, V_d)$), respectively. Therefore, based on Theorems 7 and 4, if $\forall i \in I$, r_i is valid regarding $U \prec_{lo} V$ then $U \prec_{lo} V$ since $\cup_{i \in I} r_i \cup R(U_{max}, V_{max})$ covers V_{mbb}. Consider that $V_{min} = r_{i,min}$ $(\forall i \in I)$. Immediately, each r_i is valid regarding $U \prec_{lo} V$ if

$$U.cdf(V_{min}) \geq \max_{i \in I}\{V.cdf(V_i \leq a_i)\}. \tag{1}$$

In our pruning technique, we can precisely calculate $U.cdf(V_{min})$ for both cases since $U \in R_{ssky}$ is already read in memory; and the *Cantelli's Inequality* [Meester 2004] is employed to provide an upper bound for $V.cdf(V_i \leq a_i)$. Let $\delta(x, y)$ be $\frac{1}{1+\frac{x^2}{y^2}}$ if $y \neq 0$, 1 if $x = 0$ and $y = 0$, and 0 if $x \neq 0$ and $y = 0$.

THEOREM 15 (CANTELLI'S INEQUALITY [MEESTER 2004]). *Suppose that t is a random variable in R^1 with mean $\mu(t)$ and variance $\sigma^2(t)$, $Prob(t - \mu(t) \geq a) \leq \delta(a, \sigma(t))$ for any $a \geq 0$, where $Prob(t - \mu(t) \geq a)$ denotes the probability of $t - \mu(t) \geq a$.*

Note that Theorem 15 extends the original Cantelli's Inequality [Meester 2004] to cover the case when $\sigma = 0$ and/or $a = 0$. The following theorem provides an upper bound for $Prob(t \leq b)$ when $b \leq \mu$.

THEOREM 16. *Assume that $0 \leq b \leq \mu(t)$. Then, $Prob(t \leq b) \leq \delta(\mu(t) - b, \sigma(t))$.*

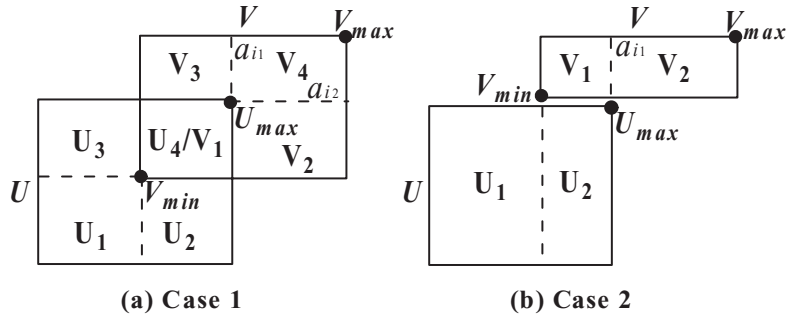

Fig. 13. Statistic pruning for gskyline.

PROOF. Let $t' = 2\mu(t) - t$. It can be immediately verified that $\sigma^2(t') = \sigma^2(t)$ and $\mu(t) = \mu(t')$. Applying Cantelli's Inequality on t', the theorem holds. □

Now we generalize the previous observations formally into our second pruning rule. Let $U_{max} = (a_1, a_2, \ldots, a_d)$, $E_{min} = (b_1, b_2, \ldots, b_d)$, and I is the subset of $\{1, \ldots, d\}$ such that $a_i > b_i$ if $i \in I$ and $a_i \leq b_i$ otherwise. Let $\Delta(E, U) = \max_{i \in I}\{\delta(\mu_i(E) - a_i, \sigma_i(E))\}$ if $a_i \leq \mu_i(E)$ for each $i \in I$; and $\Delta(E, U) = \infty$ otherwise.

PRUNING RULE 2. *Suppose that $U_{min} \preceq E_{min}$ and $U_{max} \prec E_{max}$. If $U.cdf(E_{min}) \geq \Delta(E, U)$, then every object in the entry E of the global R-tree is stochastically dominated by U regarding the lower orthant order; that is, E can be pruned by U.*

The proof of Pruning Rule 2 is presented in Appendix A.4.

5.4. Analysis of Algorithm 4 for Computing lskyline

Prune(R_{ssky}, E) in Algorithm 4 is conducted as follows. For each object U in R_{ssky}, we first check Pruning Rule 1 and then Pruning Rule 2. It immediately terminates and returns true if E is pruned. Clearly, Pruning Rule 1 runs in time $O(d)$ and Pruning Rule 2 runs in time $O(d + T(U_{artree}))$ for each $U \in R_{ssky}$.

Correctness. Based on the correctness of our verification algorithm in Section 3.3, and Pruning Rules 1 and 2, it can be immediately shown that Algorithm 4 is correct.

Access Order of R_{ssky}. In Algorithm 4, the objects in R_{ssky} can be accessed in any order. Nevertheless, in our implementation, we access objects U in R_{ssky} according to the increasing order of $dist(U_{min})$ with the aim to maximize the chance that Verification(U, V) may terminate earlier and an entry may be pruned earlier.

5.5. Filtering in Computing gskyline

According to Definition 4, it is immediate that Pruning Rule 1 in computing lskyline, the MBB-based Pruning in Section 5.3, can also be used as the first pruning rule in computing gskyline. Nevertheless, Pruning Rule 2 in computing lskyline is not applicable since $U \prec_{lo} V$ does not imply $U \prec_{uo} V$. To follow is a new statistic-aggregate-based pruning rule in computing gskyline.

Pruning Based on Statistic Aggregate. Suppose that E cannot be pruned by U by Pruning Rule 1; that is, $U_{max} \not\preceq E_{min}$. Intuitively, E could still be pruned if U_{max} is very "close" to E_{min}. Assuming E is an object V, the basic idea is to iteratively "trim" V by U using Pruning Rule 1; if V can be completely trimmed off, then $U \prec_{uo} V$ (i.e., V is pruned). What follows is an example.

Consider U and V in Figure 13(a). Clearly, U_1 fully dominates $V_1 \cup V_2$ and $V_1 \cup V_3$, respectively. If $U.cdf(U_1) \geq V.cdf(V_1 \cup V_2)$, then $V_1 \cup V_2$ can be trimmed away from V.

Iteratively, since $U_1 \cup U_3$ fully dominates V_3, V_3 can be trimmed away if $U.cdf(U_1 \cup U_3) - V.cdf(V_1 \cup V_2) \geq V.cdf(V_3)$, where $U.cdf(U_1 \cup U_3) - V.cdf(V_1 \cup V_2)$ is the probability mass of U in $U_1 \cup U_3$ after discounting the trimming of $V_1 \cup V_2$. Since V_4 is fully dominated by U and $U.cdf(U) = V.cdf(V) = 1$, we can claim $U \prec_{uo} V$ if $V_1 \cup V_2 \cup V_3$ is trimmed away. Similarly, for another possible trimming permutation, we need to test if $U.cdf(U_1) \geq V.cdf(V_1 \cup V_3)$ and $U.cdf(U_1 \cup U_2) - V.cdf(V_1 \cup V_3) \geq V.cdf(V_2)$ to claim $U \prec_{uo} V$. Pruning Rule 3 shortly formally presents this observation in R^d_+ based on using Cantelli's Inequality to provide an upper bound of the probability mass (see Section 5.3).

We need the following notations to present Pruning Rule 3. Given U and V ($U \neq V$), let $U_{min} \preceq V_{min}$ and $U_{max} \prec V_{max}$ where $U_{max} = (a_1, a_2, \ldots, a_d)$ and $V_{min} = (b_1, b_2, \ldots, b_d)$. Assume that I ($\neq \emptyset$) is the subset of $\{1, \ldots, d\}$ such that $a_i > b_i$ for $i \in I$ and $a_i \leq b_i$ otherwise; note $I = \emptyset$ implies $U \prec_{uo} V$ based on Pruning Rule 1.

Let $|I| = k$, and $\{i_1, i_2, \ldots, i_k\}$ is a permutation of I. For $0 \leq j \leq k-1$, $R^U_{i_j, \leq b}$ denotes the set of points $(x_1, \ldots, x_d)$ in U_{mbb} such that $x_i \leq b_i$ if $i \in \{i_{j+1}, \ldots, i_k\}$; that is, the remaining of U_{mbb} after its "right" parts are cut by the $(k-j)$ hype-planes $x_i = b_i$ for $i \in \{i_{j+1}, \ldots, i_k\}$, respectively; note that each point in $R^U_{i_0, \leq b}$ fully dominates V_{mbb}.

Example 11. Regarding the example in Figure 13, assume that the horizontal dimension is 1 and the vertical dimension is 2. Clearly, $I = \{1, 2\}$ for the two objects U and V in Figure 13(a); and $I = \{1\}$ for the example in Figure 13(b).

Regarding Figure 13(a), if $(1, 2)$ is a permutation of I (i.e., $i_1 = 1$ and $i_2 = 2$), $R^U_{i_0, \leq b}$ is U_1, $R^U_{i_1, \leq b}$ is $U_1 \cup U_2$.

Regarding Figure 13(b), as I only contains the dimension 1, there is only one permutation; that is, $i_1 = 1$. $R^U_{i_0, \leq b}$ is U_1.

Given a $U \in R_{ssky}$ and an entry E in sR-tree, let $U_{min} \preceq E_{min}$, and $U_{max} \prec E_{max}$ where $U_{max} = (a_1, a_2, \ldots, a_d)$ and $E_{min} = (b_1, b_2, \ldots, b_d)$. For $1 \leq i \leq d$, $\Delta_i(E, U) = \delta(\mu_i(E) - a_i, \sigma_i^2(E))$ when $a_i \leq \mu_i(E)$ and $\Delta_i(E, U) = \infty$ otherwise.

PRUNING RULE 3. *Assume $U_{min} \preceq E_{min}$, $U_{max} \prec E_{max}$, and I is the subset of $\{1, \ldots, d\}$ such that $a_i > b_i$ for $i \in I$ and $a_i \leq b_i$ otherwise. Suppose that $\{i_1, \ldots, i_k\}$ is a permutation of I. If the following conditions hold, then for every object V in E, $U \prec_{lo} V$; that is, E can be pruned by U.*

(1) *For each $i \in I$, $a_i < \mu_i(E)$.*
(2) *For $1 \leq j \leq k$, $U.cdf(R^U_{i_{j-1}, \leq b}) \geq \sum_{l=1}^{j} \Delta_{i_l}(E, U)$.*

The proof of Pruning Rule 3 is presented in Appendix A.5.

5.6. Execution and Analysis of Algorithm 4 in Computing gskyline

Prune(R_{ssky}, E) in Algorithm 4 is conducted as follows. For each entry E and each object U in R_{ssky}, we first check Pruning Rule 1 and then check Pruning Rule 3. It terminates and returns true if E is pruned. Pruning Rule 1 runs in time $O(d)$.

If a permutation of I is given, testing the conditions in Pruning Rule 2 can be conducted in $O(d + k \times t(U_{rtree}))$ where $O(t(U_{rtree}))$ is the average time complexity to run the window aggregate techniques in Tao and Papadias [2012] to calculate $U.cdf$s and $k = |I|$. In our implementation, we examine one particular permutation obtained by the greedy paradigm such that iteratively, for $1 \leq j \leq k$, we choose i_j from I such that $(U.cdf(R^U_{i_{j-1}, \leq b}) - \sum_{l=1}^{j} \Delta_{i_l}(E, U))$ is maximized and the pruning fails/terminates immediately if it is negative. The complexity is $O(d + k^2 t(U_{rtree}))$ for each $U \in R_{ssky}$ and each entry E.

Note that using the permutation, generated by the greedy heuristic, may fail to find a "feasible" permutation to meet the conditions in Pruning Rule 2. Nevertheless, our initial experiment in a lower-dimensional space ($d \leq 4$) shows that possible gains in pruning powers are very small by exploring all permutations. We omit a further improvement due to space limits.

Correctness. Based on the correctness of our verification algorithms (Section 4), and Pruning Rules 1 and 3, Algorithm 4 for computing gskyline is correct.

Access Order of R_{ssky}. We use the same access order as that in computing lskyline with the same aim.

6. EMPIRICAL STUDY

We conduct a thorough performance evaluation on the efficiency and effectiveness of our techniques to compute both lskyline and gskyline. Since this is the first work in stochastic skyline computation, the performance evaluation is conducted against our techniques only. We implement the following techniques.

—*lsky*. Algorithm 4 proposed in Section 5.2, together with the verification algorithm (Algorithm 1) in Section 3.3 and Pruning Rules 1 and 2 in Section 5.3, to compute lskyline.
—*lsky-NoP*. *lsky* without Pruning Rule 1 and 2.
—*lsky-NoP2*. *lsky* without Pruning Rule 2.
—*lsky-NaiveV*. *lsky* with the filtering techniques in Section 5.3 and the naive verification algorithm in Section 4.1.
—*gsky*. Algorithm 4 proposed in Section 5.2, together with the verification algorithm (QuickTest) in Section 4.4 and the two pruning rules (Pruning Rules 1 and 3) in Section 5.5, to compute gskyline.
—*gsky-NoP*. *gsky* without two pruning rules - Pruning Rules 1 and 3.
—*gsky-NoP3*. *gsky* without the second pruning rule in computing gskyline- Pruning Rule 3.
—*gsky-BasicV*. *gsky* with the filtering techniques in Section 5.5 and the basic verification algorithm Algorithm 2 in Section 4.3.

6.1. Experiment Setup

All algorithms proposed in the article are implemented in standard C++ with STL library support and compiled with GNU GCC. Experiments are conducted on a PC with Intel Xeon 2.4 GHz dual CPU and 4G memory under Debian Linux. In our implementation, MBBs of the uncertain objects are indexed by an sR-tree with page size 4096 bytes. The instances of an object are organized by a main-memory-based aggregate R-tree with fan-out 4 and we load in the whole aggregate R-tree of V if V cannot be pruned by the filtering techniques. The open source-code of the max-flow algorithm in Hochbaum [2008] is used in our algorithm.

We use both real and synthetic datasets in our evaluation process.

The real dataset is extracted from NBA players' game-by-game statistics from 1991 to 2005 (http://www.nba.com), containing $339,621$ records of $1,313$ players. A player is treated as an uncertain object with the game records as its instances. Instances of an object take equal probability. Three attributes are selected in our experiments: the number of points, the number of assistances, and the number of rebounds. The NBA dataset is employed since the MBBs of players have a very large overlapping degree; thus it may give a good challenge to our techniques.

Synthetic data is generated by modifying the benchmark data generator in Börzsönyi et al. [2001] with the following parameters. The centers of object MBBs

Table III. Parameters

edge length h	0.02, **0.04**, 0.06, 0.08, 0.1
dimensionality d	2, **3**, 4, 5
number of objects n	**20k**, 40k, 60k, 80k, 100k
number of instances m	200, **400**, 600, 800, 1k
object center distribution	**anti**, corr, inde
instance distribution	unif, zipf, norm, **mix**

Table IV. pskyline vs Stochastic Skylines

p	# psky	# hit of lskyline	# hit of gskyline
0.5	0	0	0
0.05	83	76	80
0.005	326	122	128
5×10^{-10}	837	124	131

(a) NBA(# lskyline: 127, # gskyline: 135)

p	# psky	# hit of lskyline	# hit of gskyline
0.8	3	3	3
0.08	190	146	152
0.008	427	201	217
3×10^{-10}	1135	264	283

(b) 3d (# lskyline: 283, # gskyline: 303)

follow *Anti-correlated (anti)*, *Correlated (corr)*, or *Independent (inde)* distributions with the default **anti**. Each dimension value domain is $(0, 1]$. The MBBs of objects are hyper-cube and their edge length randomly varies in $(0, 2h]$ where the average length h varies from 0.02 to 0.1 with the default **0.04**. The average number m of instances in each object varies from 200 to 1000 with the default **400**. Instance locations of an object follow one of the 4 distributions, $uniform(unif)$, $normal(norm)$, $zipf$, or a mixture of $unif$, $norm$ and $zipf(mix)$. In $unif$, the instances of an object are distributed uniformly inside an MBB with the same occurrence probability. In $zipf$, one instance u of an object is first randomly allocated and then the distances from all other instances to u follow the $zipf$ distribution with $z = 0.5$ and the instance occurrence probabilities also follow the same $zipf$ distribution. In $norm$, the instances follow the normal distribution inside MBB with the standard deviation $\sigma = 0.2 \times h$, and the occurrence probabilities are the same. In mix, the preceding 3 distributions are mixed and each distribution has a portion of $\frac{1}{3}$; **mix** is used as the default in the experiment.

We also study the impact of other key parameters. Due to space limits, some experiment results are presented in Appendix B. The dimensionality (d) varies from 2 to 5 with the default value **3**. The number n of objects varies from $20k$ to $100k$ where the default value is 20k.

Table III summarizes parameter ranges and default values (in bold font). Note that in the default setting, the total number of instances is 8 millions. The maximal number of total instances of the datasets is 40 millions. In the experiments that fellow, these parameters use default values unless otherwise specified.

6.2. Size of Stochastic Skyline

pskyline, lskyline, and gskyline. We use the NBA data and a $3d$ synthetic dataset with the centers of MBBs following the inde distributions. The experiment result in Table IV shows the number of probabilistic skyline objects [Pei et al. 2007] (# psky), the number of lskyline objects (# lskyline), and the number of gskyline objects (# gskyline) where p indicates the threshold used in generating probabilistic skylines, and # hit of lskyline and # hit of gskyline are the numbers of lskyline objects and gskyline objects contained in the corresponding probabilistic skyline, respectively. It demonstrates that a non-stochastic skyline object may have a quite large skyline probability, while a stochastic skyline object may have very small skyline probability; that is, the probabilistic skyline model is orthogonal to the stochastic skyline models.

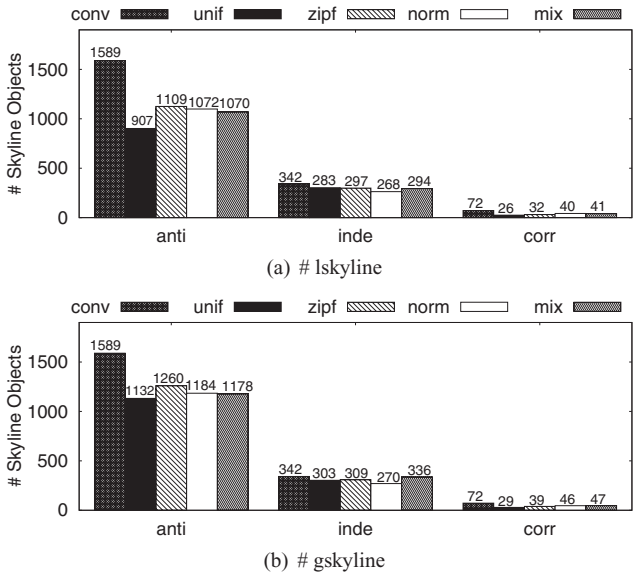

Fig. 14. Skyline size vs different distributions.

Note that gskyline is a superset of lskyline (Section 2.2); the experiment also demonstrates this. In our experiment, to retain the preference of the smaller values, we change all statistics to negative values. We use the binary code in Pei et al. [2007] to generate probabilistic skylines.

Figure 14(a) and Figure 14(b) report the sizes of lskyline and gskyline regarding different distributions of instances locations and object MBB centers, respectively. The x-coordinate in Figure 14 indicates different distributions of MBB centers. For each distribution of MBB centers, we record the number of gskyline objects regarding a distribution of instance locations. *conv* denotes the conventional skyline over the data points in a $4d$-space where each data point is the combination of the lower corner and the edge length of an MBB. It demonstrates that the sizes of lskyline and gskyline may be bounded by such *conv* in practice.

We further explore the size difference of lskyline and gskyline regarding the number of objects, average number of instances per object, and the distribution of object MBBs' centers. The results show that the difference is small in practice though theoretically the ratio of #gskyline over #lskyline may equal the number of all uncertain objects. Details are presented in Appendix B.1.

6.3. Evaluating Efficiency

We evaluate the performance of *lsky*, *lsky-NoP2*, *gsky* and *gsky-NoP3* since the filtering techniques combining naive verification algorithms (i.e., *lsky-NaiveV* and *gsky-BasicV*) or the algorithms without filtering techniques (i.e., *lsky-NoP* and *gsky-NoP*) are very inefficient (details are reported in Appendix B.2).

Figure 15 reports the evaluation of *lsky*, *lsky-NoP2*, *gsky*, and *gsky-NoP3* against different distributions of instance locations and NBA data. Both *lsky* and *gsky* are quite efficient. It also shows that *lsky* and *gsky* always significantly outperform *lsky-NoP2* and *gsky-NoP3*, respectively; that is, the Pruning Rule 2 and Pruning Rule 3 are very effective in practice. Note that *lsky* is significantly more efficient than *gsky* when the dimensionality is low, while the Appendix shows that *gsky* beats *lsky*

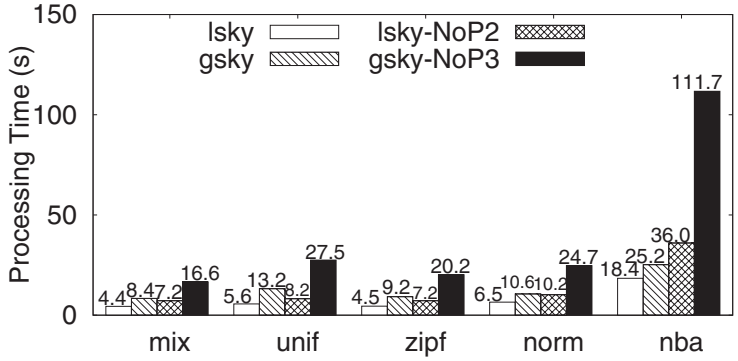

Fig. 15. Processing time regarding different distributions.

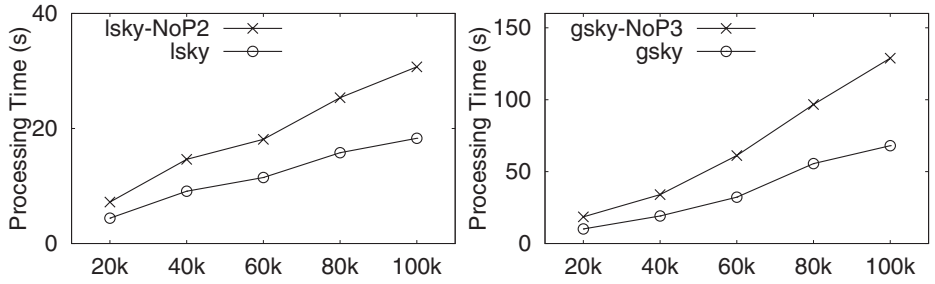

Fig. 16. Processing time regarding #Objects.

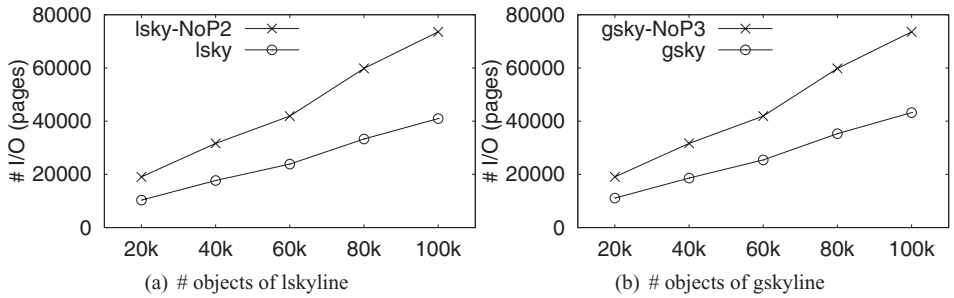

Fig. 17. I/O costs regarding different parameters.

when the dimensionality increases to 7 due to the NP-completeness regarding the dimensionality.

In Figure 16, we evaluate the scalability of our algorithms against different dataset sizes. Figure 16(a) and (b) show that the performance of all algorithms degrades "linearly" with the growth of dataset size. The experiment is also conducted while varying the number of instances, MBB sizes, and dimensionality; the results are presented in Appendix B.2 and show that computing gskyline is more expensive than lskyline.

We further examine the effectiveness of the statistic pruning rules (i.e., Pruning Rule 2 and Pruning Rule 3) by evaluating the I/O costs reduced. Figure 17 reports the number of pages accessed in *lsky*, *lsky-NoP2*, *gsky*, and *gsky-NoP3* regarding dataset sizes. It shows that the Pruning Rule 2 and Pruning Rule 3 are very effective and lead

Table V. Time in Seconds (s)

	20k	40k	60k	80k	100k
Filter	0.27	0.67	0.82	1.31	1.39
Verify	1.93	3.56	5.56	7.47	4.48
I/O	2.18	4.47	4.98	6.89	8.26

(a) *lsky*

	20k	40k	60k	80k	100k
Filter	0.46	1.34	1.81	3.60	4.70
Verify	5.70	14.34	20.80	38.20	45.84
I/O	2.27	4.67	5.19	7.15	8.58

(b) *gsky*

to around 2 times reduction of the number of pages accessed. A similar trend of I/O costs is observed by varying dimensionality and we report the details in Appendix B.2.

Table V gives a breakdown information of the filtering, verification, and I/O time (in seconds). The results are reported where we vary the number of objects from $20k$ to $100k$. The results show that the filtering costs are much smaller than the costs of verification and I/O.

Note that Pruning Rule 3 can also be used in *lsky* instead of Pruning Rule 2. As expected, Pruning Rule 2 is more effective than Pruning Rule 3. Details of our experiment results are presented in Appendix B.2.

7. RELATED WORK

Conventional Skyline Computation. Skyline computation over conventional (certain) data has a long history. Börzsönyi et al. [2001] are the first to study the problem of computing skylines in the context of databases and propose an SQL syntax for the skyline query. They develop two computation techniques based on *block nested loop* (BNL) and *divide-and-conquer*, respectively. Another BNL-based technique SFS (*sort filter skyline*) is proposed in Chomicki et al. [2003] with the aim to improve BNL by sorting a dataset first. SFS is then significantly improved by LESS (*linear elimination sort for skyline*) in Godfrey et al. [2005]. *Sort and limit skyline algorithm* (SaLSa) [Bartolini et al. 2008] aims to improve SFS or LESS by avoiding scanning the complete set of sorted objects.

The index-based progressive techniques are firstly proposed in Tan et al. [2001] where two data structures, *Bitmap* and *Index*, are proposed. Kossmann et al. [2002] present another progressive technique based on the nearest-neighbor search technique. A *branch-and-bound* (BBS) algorithm to progressively output skyline points based on R-trees with the guarantee of minimal I/O cost is developed in Papadias et al. [2003]. Various advanced index-based techniques are recently developed (e.g., Lee et al. [2007] and Zhang et al. [2009]).

Variations of skyline computation have also been extensively explored; for example, skylines for partially ordered value domains[Chan et al. 2005] and skyline cubes [Pei et al. 2006].

Skyline Computation over Uncertain Data. Considerable research effort has been put into modeling and managing uncertain data in recent years due to many emerging important applications (e.g., Boulos et al. [2005] and Sarma et al. [2006]).

Probabilistic skyline on uncertain data is first tackled in Pei et al. [2007] where skyline objects are retrieved based on skyline probabilities. Efficient techniques are proposed following the bounding-pruning-refining framework. Lian and Chen [2008] combine reverse skyline with uncertain semantics and study the probabilistic reverse skyline problem in both monochromatic and bichromatic fashion. In Atallah et al. [2011], novel, subquadratic algorithms are developed to compute skyline probabilities of every object. A stochastic skyline operator, lskyline, is studied in a preliminary version [Lin et al. 2011] of this article; nevertheless, lskyline only provides the minimum candidate set for all multiplicative decreasing functions. This article is a significant extension of the preliminary version [Lin et al. 2011] by including our

investigation of gskyline which gives the minimum candidate set for all decreasing ranking functions.

8. CONCLUSION

In this article, we introduce a novel *stochastic skyline* model on uncertain data with the aim to provide a minimum set of candidates to the optimal solutions over the family of multiplicative decreasing scoring functions, as well as general decreasing scoring functions, for users to make their personal trade-offs. We investigate the problem of computing stochastic skyline regarding the lower orthant order and the usual order, namely lskyline and gskyline. For the first time in the literature, we show that the problem of determining the lower orthant order is NP-complete regarding the dimensionality; consequently lskyline is NP-complete in the same sense. Then we show that while the lower orthant order is much simpler than the usual order in terms of geometry, determining the usual order can be solved in polynomial time in contrast to the complexity of determining the lower orthant order. We also develop novel and efficient polynomial-time algorithms to compute the stochastic skyline regarding the lower orthant order and usual order, respectively, over large datasets. Comprehensive experiments are conducted on both real and synthetic data to demonstrate the efficiency of our techniques.

Possible Future Work. In the article, we focus on discrete cases of PDF. The framework and the pruning rules work for continuous PDF in principle; nevertheless the calculation of probability mass involves integration which is expensive. Moreover, verification is a big challenge for usual order between two uncertain objects, while the partitioning paradigm in Section 3.3 works for determining lskyline though it may involve an infinite number of partitions. As an alternative, we can discretize a continuous PDF by sampling methods. While the framework is immediately applicable to the sampled points, the main issue is to estimate the accuracy of a sampling method. This is one possible future work.

Specifying stochastic skylines regarding other measures/variance is nontrivial; this could be a very interesting future work. Another possible future work is to compute lskyline and gskyline over any subspace [Pei et al. 2006]. While our verification techniques between two objects are immediately applicable, developing effective index techniques to support any subspace computation is a challenge. The next question is how to compute representative stochastic skylines [Lin et al. 2007].

Finally, while our work in the article deals with independent uncertain objects, uncertain objects are correlated in many applications. It is a challenge to specify stochastic skylines against correlated data regarding the expected utility principle.

REFERENCES

ATALLAH, M. J., QI, Y., AND YUAN, H. 2011. Asymptotically efficient algorithms for skyline probabilities of uncertain data. *ACM Trans. Datab. Syst. 32*, 2, 12.

BARTOLINI, I., CIACCIA, P., AND PATELLA, M. 2008. Efficient sort-based skyline evaluation. *ACM Trans. Datab. Syst. 33*, 4.

BÖRZSÖNYI, S., KOSSMANN, D., AND STOCKER, K. 2001. The skyline operator. In *Proceedings of the International Conference on Data Engineering (ICDE'01).* 421–430.

BOULOS, J., DALVI, N., MANDHANI, B., MATHUR, S., RE, C., AND SUCIU, D. 2005. MYSTIQ: A system for finding more answers by using probabilities. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. 891–893.

BRINKHOFF, T., KRIEGEL, H.-P., AND SEEGER, B. 1993. Efficient processing of spatial joins using r-trees. In *Proceedings of the ACM SIGMOD Conference on Management of Data.* 237–246.

CHAN, C.-Y., ENG, P.-K., AND TAN, K.-L. 2005. Stratified computation of skylines with partially ordered domains. In *Proceedings of the ACM SIGMOD Conference on Management of Data.* 203–214.

CHAUDHURI, S., DALVI, N. N., AND KAUSHIK, R. 2006. Robust cardinality and cost estimation for skyline operator. In *Proceedings of the International Conference on Data Engineering (ICDE'06)*. 64.

CHOMICKI, J., GODFREY, P., GRYZ, J., AND LIANG, D. 2003. Skyline with presorting. In *Proceedings of the International Conference on Data Engineering (ICDE'03)*. 717–719.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms* 2nd Ed. The MIT Press, Cambridge, MA.

GAREY, M. AND JOHNSON, D. 1990. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

GODFREY, P., SHIPLEY, R., AND GRYZ, J. 2005. Maximal vector computation in large data sets. In *Proceedings of the International Conference on Very Large Databases (VLDB'05)*. 229–240.

HOCHBAUM, D. 2008. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Oper. Res. 56*, 4, 992–1009.

KIJIMA, M. AND OHNISHI, M. 1999. Stochastic orders and their applications in financial optimization. *Math. Methods Oper. Res. 50*, 2, 351–372.

KOSSMANN, D., RAMSAK, F., AND ROST, S. 2002. Shooting stars in the sky: An online algorithm for skyline queries. In *Proceedings of the International Conference on Very Large Databases (VLDB'02)*.

LEE, K. C. K., ZHENG, B., LI, H., AND LEE, W. C. 2007. Approaching the skyline in z order. In *Proceedings of the International Conference on Very Large Databases (VLDB'07)*. 279–290.

LEVY, H. 1992. Stochastic dominance and expected utility: Survey and analysis. *Manag. Sci. 38*, 4, 555–593.

LIAN, X. AND CHEN, L. 2008. Monochromatic and bichromatic reverse skyline search over uncertain databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. 213–226.

LIN, X., YUAN, Y., ZHANG, Q., AND ZHANG, Y. 2007. Selecting stars: The k most representative skyline operator. In *Proceedings of the International Conference on Data Engineering (ICDE'07)*. 86–95.

LIN, X., ZHANG, Y., ZHANG, W., AND CHEEMA, M. A. 2011. Stochastic skyline operator. In *Proceedings of the International Conference on Data Engineering (ICDE'11)*. 721–732.

MEESTER, R. 2004. *A Natural Introduction to Probability Theory*. Birkhauser.

PAPADIAS, D., TAO, Y., FU, G., AND SEEGER, B. 2003. An optimal progressive algorithm for skyline queries. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. 467–478.

PEI, J., JIANG, B., LIN, X., AND YUAN, Y. 2007. Probabilistic skylines on uncertain data. In *Proceedings of the International Conference on Very Large Databases (VLDB'07)*. 15–26.

PEI, J., YUAN, Y., LIN, X., JIN, W., ESTER, M., LIU, Q., WANG, W., TAO, Y., YU, J. X., AND ZHANG, Q. 2006. Towards multidimensional subspace skyline analysis. *ACM Trans. Datab. Syst. 31*, 4, 1335–1381.

SARMA, A. D., BENHELLOUN, O., HALEVY, A., AND WIDOM, J. 2006. Working models for uncertain data. In *Proceedings of the International Conference on Data Engineering (ICDE'06)*. 7.

SHAKED, M. AND SHANTHIKUMAR, J. G. 2007. *Stochastic Orders and Their Applications*. Academic Press.

STEUER, R. E. 1995. *Multi Criteria Optimization: Theory, Computation, and Application*. John Wiley and Sons.

TAN, K.-L., ENG, P.-K., AND OOI, B. C. 2001. Efficient progressive skyline computation. In *Proceedings of the International Conference on Very Large Databases (VLDB'01)*. 301–310.

TAO, Y. AND PAPADIAS, D. 2012. Range aggregate processing in spatial databases. *IEEE Trans. Knowl. Data Engin. 16*, 12.

ZHANG, S., MAMOULIS, N., AND CHEUNG, B. W. 2009. Scalable skyline computation using object-based space partitioning. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. 483–494.

Received August 2011; revised January 2012; accepted March 2012