Stochastic Skyline Operator

Xuemin Lin Ying Zhang Wenjie Zhang Muhammad Aamir Cheema

University of New South Wales, Australia
{lxue, yingz, zhangw, macheema}@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-1020
October 2010

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

In many applications involving the multiple criteria optimal decision making, users may often want to make a personal trade-off among all optimal solutions. As a key feature, the skyline in a multi-dimensional 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 mater how their preferences vary. Driven by many 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 paper we propose a novel skyline operator, namely stochastic skyline. In the light of the expected utility principle, stochastic skyline guarantees to provide the minimum set of candidates for the optimal solutions over all possible monotonic utility functions. In contrast to the conventional skyline or the probabilistic skyline computation, we show that the problem of stochastic skyline is NP-complete with respect to the dimensionality. Novel and efficient algorithms are developed to efficiently compute stochastic skyline over multi-dimensional uncertain data, which run in polynomial time if the dimensionality is fixed. We also show, by theoretical analysis and experiments, that the size of stochastic skyline is 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.

1 introduction

In a d-dimensional space R^d, the *skyline* is defined over the given preferences of coordinate values on each dimension (i.e., either smaller or larger coordinate values are preferred). Given two points x and y in R^d, x *dominates* y if x is not worse than y on each dimension and is better than y on one dimension according to the given preferences of coordinate values. Given a set D of points (objects) in R^d, the preferences across different dimensions may conflict to each other regarding D; that is, D does not always contain a point that dominates every other point in D. Consequently, a scoring function f is often required to rank the points in D to capture the preferences on each dimension to generate the optimal decision (solution). Without loss of generality, *in the rest of paper we assume smaller coordinate values are preferred on each dimension and all points are in R^d_+ (i.e. coordinate values are non-negative)*. This implies that to capture such preferences on each dimension and make the optimal solution with the maximum score, a scoring function f required to rank objects in D should be decreasing.

The *skyline* of D consists of the points in D which are not dominated by any other points in D. It is well known [4, 22] that x dominates y if and only if for any decreasing multivariate function f, $f(x) \geq f(y)$; this is formally stated in page 266 of [21] if a point is regarded as an uncertain object with only one instance that has the occurrence probability 1. Therefore, in multi-criteria decision applications when users are not content with being given only one optimal solution and want to make a personal trade-off among different optimal solutions (i.e., with maximal scores) over all possible decreasing scoring functions, the skyline provides the *minimal* set of candidates by removing the points not preferred by any decreasing scoring functions (users). Skyline computation over certain data has been extensively studied (e.g., [4, 17, 25]).

Probabilistic Skyline. 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 [18] 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 [18], efficient techniques are developed to retrieve uncertain objects with skyline probabilities greater than a given threshold, while [2] provides efficient techniques to compute skyline probabilities for all objects.

Motivation. The research towards the problem of multiple criteria optimization over uncertain objects has a long history in economics, finance, and mathematics; see [11, 14, 21] for example. The *expected utility principle* is the most popular model [11, 14] to select the optimal uncertain object against multiple criteria. In the light of the expected utility principle, 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.

Assume that a head coach wants to select the best high-jump athlete from all available 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* trails 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 distribution. For this purpose, an athlete U may be treated as an uncertain object and her performance at 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 1.1 gives a small scale example where 3 players are involved.

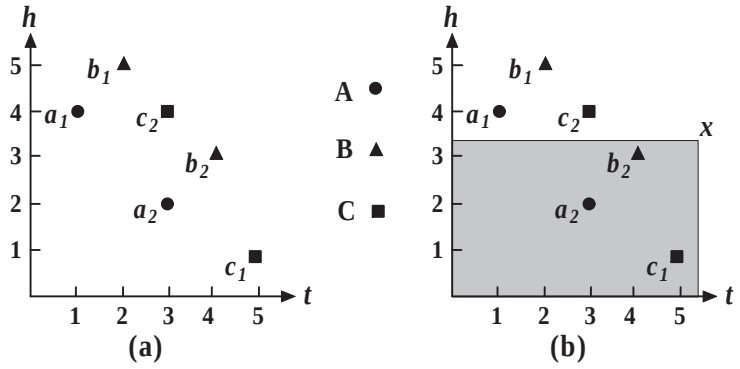


Figure 1.1: Motivating Example

As depicted in Figure 1.1(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 tradeoffs 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. If the coach is taking more risks and only wants to select the athlete with a chance to smash the world record, 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^{-bt}. If these utility functions are used, then C is the optimal solution since $E[f(A)]$ and $E[f(B)]$ are 0. This example shows that B is never preferred 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.

While very useful in applications to determine probabilistic dominance relation-

ships, skyline probabilities cannot capture the preferences of utility (scoring) functions regarding the expected utility principle. Regarding the example in Figure 1.1(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 players based on skyline probability value, then B is always preferred to C; that is, there is no chance to exclude the object B without excluding the object C. This is an inherent limitation of the probabilistic skyline model.

Stochastic Skyline. Stochastic orders have been widely used in many real-life applications [11, 14, 21], including economics, finance, and multi-criteria statistic decision making. Generally, given a class $\mathcal{F}$ 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 [11]); that is, all users prefer U to V. Given a set $\mathcal{U}$ of uncertain objects, the *stochastic order based skyline*, namely stochastic skyline regarding $\mathcal{F}$, is the subset of $\mathcal{U}$ such that each object U in the stochastic skyline is not stochastically dominated by any other object in $\mathcal{U}$ regarding $\mathcal{F}$. Consequently, in applications of multiple criteria optimal decision making over uncertain objects, the stochastic skyline regarding $\mathcal{F}$ provides the minimum set of candidates for the optimal solutions (maximum values), respectively, for all functions in $\mathcal{F}$ by removing the objects not preferred by any function in $\mathcal{F}$.

Several stochastic orders have been defined and their mathematic properties have been well studied in the statistic literature [11, 21]. Consider that multiplicative non-negative 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 non-negative decreasing. In this paper, we investigate the problem of efficiently computing the stochastic skyline regarding *lower orthant order* [21] since the lower orthant order $\prec_{\mathcal{F}}$ is defined over the family $\mathcal{F}$ of multiplicative decreasing functions. Intuitively, an uncertain object U stochastically dominates V regarding the lower orthant order if and only if for any point x in R_+^d, the probability mass of U is not smaller than the probability mass of V in the region "dominated" by x (i.e. the shaded region in Figure 1.1(b)); this will be formally defined in Section 2. Note that as expected, such stochastic skyline excludes object B and includes A and C in the example in Figure 1.1(a).

Challenges and Contributions. A main challenge for computing stochastic skyline is how to efficiently check the stochastic dominance relationship between two uncertain objects U and V. A naive way to check every point in the whole multi-dimension data pace involves an infinite number of points; thus it is computationally infeasible. Efficiently computing stochastic skyline among a large number of objects is even harder. In the paper, following the *filtering-verification* paradigm we develop novel spatial- and statistics-based filtering techniques to efficiently and effectively eliminate non-promising uncertain objects. Our filtering techniques are based on an R-Tree. We also develop an efficient verification algorithm that only needs to check a finite number of points to verify whether U stochastically dominates V.

To the best of our knowledge, this is the first attempt to introduce the stochastic skyline model over uncertain data. Consider that in many applications, probability density functions (PDFs) of an uncertain object are often described by a set of instances and their occurrence probabilities. In this paper, our investigation focuses on discrete cases of PDFs. 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 over the family

of multiplicative decreasing scoring functions for users to make their personal trade-offs.

- We show that the problem of determining if an uncertain object is stochastically dominated by another uncertain object is NP-complete regarding the dimensionality, in contrast to the conventional skyline and probabilistic skyline.
- We develop a novel, efficient algorithm to verify if an uncertain object is stochastically dominated by another object. The algorithm runs in polynomial time if the dimensionality is fixed.
- We propose effective and efficient filtering techniques to reduce the number of verifications in computing stochastic skyline.
- We also show, by theoretical analysis and experiments, that the size of stochastic skyline is quite similar to that of skyline over certain data.

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 paper is organized as follows. In Section 2, we formally define the problem and present preliminaries. In Section 3, we show that the problem of determining the stochastic order between two objects is NP-complete regarding the dimensionality d and then present an efficient verification algorithm that runs in polynomial time if d is fixed. Then we present our framework, novel filtering techniques, and a size estimation of stochastic skyline in Section 4, as well as discussions of other stochastic orders and continuous cases of PDFs. Experiment results are presented in Section 5. Section 6 provides the related work and Section 7 concludes the paper.

2 Background Information

Table 2.1 below summarizes the mathematical notations used throughout this paper.

Notation	Definition
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
$U.cdf$	the cumulative distribution function of U
$\mu(U)$	the *mean* of U
$\sigma_i^2(U)$	the *variance* of U on the i-th dimension
$U \prec_{sd} V$	U stochastically dominates V

Table 2.1: The summary of notations.

2.1 Problem Definition

A point (instance) x referred in the paper, by default, is in R_+^d where R_+^d consists of the points in R^d with nonnegative coordinate values. Without loss of generality, *in the paper we only consider the R_+^d space*.

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 paper 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 $\sum_{i=1}^{m} p_{u_i} = 1$. We assume that $m \geq 2$ since $m = 1$ means a certain object.

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 $R((0, ..., 0), x)$ where $(0, 0, ..., 0)$ is the origin in R^d; that is, $U.cdf(x) = \sum_{u \preceq x, u \in U} p_u$.

The minimal bounding box U_{mbb} of U is the minimal rectangular region r such that $U.cdf(r) = 1$. For presentation simplicity, uncertain objects are hereafter abbreviated to "objects", and we may abuse the notation of U by using U to represent U_{mbb} whenever there is no ambiguity. Moreover, the lower (upper) corner of U_{mbb} is denoted by U_{min} (U_{max}).

Example 1 *Suppose the instances of each object in Figure 1.1(b) have the appearance probabilities as described in the example depicted in Figure 1.1(a). Then,* $A.cdf(x) = \frac{1}{2}$, $B.cdf(x) = \frac{1}{2}$ *and* $C.cdf(x) = \frac{1}{100}$.

Stochastic Order. As discussed in Section 1, in this paper we will focus on *lower orthant orders* [21].

Definition 1 (Stochastic Dominance) *Given two uncertain objects* U *and* V, U stochastically dominates *an object* V, *denoted by* $U \prec_{sd} V$, *if* $U.cdf(x) \geq V.cdf(x)$ *for any point* $x \in R_+^d$ *and* $\exists y \in R_+^d$ *such that* $U.cdf(y) > V.cdf(y)$.

Base on the definition, we have that $A \prec_{sd} B$, $B \not\prec_{sd} C$, $C \not\preceq_{sd} A$, and $A \not\preceq_{sd} C$ regarding the example in Figure 1.1.

Definition 2 (Stochastic Skyline) *Given a set of uncertain objects* $\mathcal{U}$, *an object* $U \in \mathcal{U}$ *is a stochastic skyline object if there is no object* $V \in \mathcal{U}$ *such that* $V \prec_{sd} U$. *The set of stochastic skyline objects is called the stochastic skyline of* $\mathcal{U}$.

Problem Statement. In this paper we investigate the problem of efficiently computing stochastic skyline of a set of uncertain objects.

2.2 Preliminary

We say that U *stochastically* equals V, denoted by $U =_{sd} V$, if $U.cdf(x) = V.cdf(x)$ for any point $x \in R_+^d$. Given two objects U and V, we define $U = V$ if there is a one to one mapping g from U to V such that for each instance u of U, $u = g(u)$ and $p_u = p_{g(u)}$. The following Lemma states that $=_{sd}$ 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, $U =_{sd} V$ *if and only if* $U = V$.

Minimality of stochastic skyline. The following Theorem is proved in [21] (pp 309).

Theorem 1 *Let $U = (U_1, ..., U_d)$ and $V = (V_1, ..., V_d)$ be two d-dimensional independent random vectors to describe objects U and V, respectively, where U_i and V_i are sub-variables of U and V on i-dimension. Assuming $U \neq V$, then $U \prec_{sd} V$ 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 non-negative decreasing functions where expectations exist.*

Theorem 1 implies that in the light of expected utility principle, the stochastic skyline of $\mathcal{U}$ provides the minimum set of candidates to the optimal solutions, respectively, over all multiplicative decreasing functions by removing the objects not preferred by any multiplicative decreasing function. Here, we say U is preferred to V by f if $E[f(U)] \geq E[f(V)]$.

Framework of computing stochastic skyline. Our techniques for computing stochastic skyline of a set $\mathcal{U}$ of objects follow the standard two phases' framework: filtering and verification. In the filtering phase, efficient filtering techniques are developed to effectively prune non-promising objects. In the verification phase, an efficient algorithm is developed to generate the stochastic skyline of $\mathcal{U}$. The key in the verification phase is to determine whether an uncertain object U stochastically dominates another uncertain object V.

3 Stochastic Dominance Testing

According to Lemma 1, testing if $U =_{sd} V$ is equivalent to testing if $U = V$ that can be conducted in $O(dm \log m)$ if instances are firstly sorted according to the lexigraphic 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 $U \neq V$. 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.

3.1 Testing Finite Number of Points Only

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

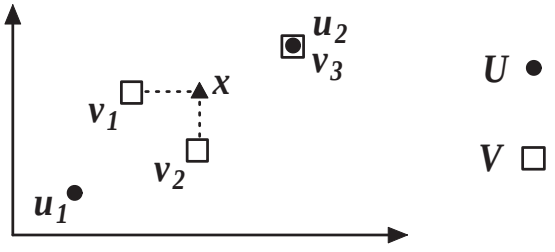


Figure 3.1: An Example

An intuitive way by checking every instance v in V to find a violation point does not work. As depicted in Figure 3.1, 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 stochastically dominates V since x in Figure 3.1 is a violation point regarding $U \prec_{sd} V$; that is, $V.cdf(x) > U.cdf(x)$.

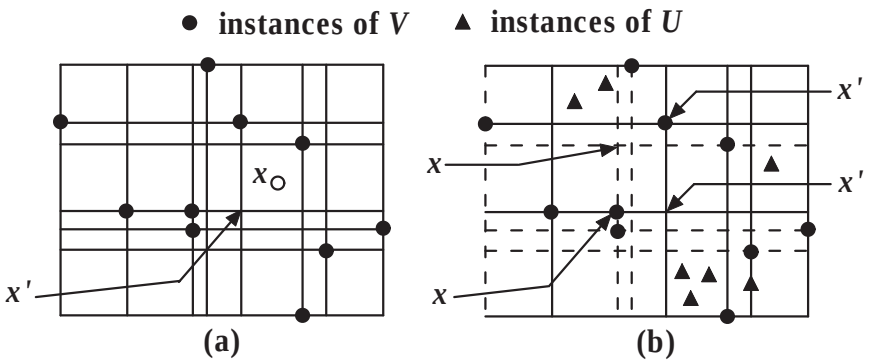


Figure 3.2: Grid Points

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 3.2(a) as an example. Theorem 2 below 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 $U \prec_{sd} V$.

Theorem 2 *Suppose that $U \neq V$. Then, V is not stochastically dominated by U if and only if there is a (grid) point $x \in \prod_{i=1}^{d} \mathcal{D}_i(V)$ that is violation point regarding $U \prec_{sd} V$ (i.e., $cdf.V(x) - cdf.U(x) > 0$).*

Proof 1 *The "if" part is immediate according to Definition 1. Below we prove the "only if" part.*

Suppose that V is not stochastically dominated by U. Then, based on Definition 1 and Lemma 1 there is a violation point regarding $U \prec_{sd} V$. It can be immediately verified that if there is a violation point regarding $U \prec_{sd} V$ 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 3.2(a) for example. It is immediate $V.cdf(x') = V.cdf(x)$ 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 $V.cdf$ and 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 2. 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

Below we show that the exponential time complexity regarding d is unavoidable.

Theorem 3 *Given two objects U and V, assume that $U \neq V$. Then, the problem of determining whether $U \not\prec_{sd} V$ is NP-complete regarding the dimensionality d.*

It is well known [9] that the minimum set cover (decision version) is NP-complete regarding d where d is the number of subsets.

Minimum Set Cover (decision version):
INSTANCE: a collection $C = \{S_i \mid 1 \leq i \leq d\}$ of subsets of S where S contains $(m+1)$ elements, positive integer $K < d$.
QUESTION: does C contain a cover for S of size K or less, i.e., a subset $C' \subseteq C$ with $|C'| \leq K$ such that every element of S belongs to at least one member (a subset of S) of C'?

Proof of Theorem 3: We convert the above minimum set cover to a special case of our problem as follows. For each instance of the minimum set cover problem, we construct an instance of our problem as follows. Let $m > 1$, $d > 1$, and $S = \{u_i \mid 0 \leq i \leq m\}$.

U has $(m+1)$ instances, u_0, u_1, ..., u_m with the occurrence probability p_i ($0 \leq i \leq m$), respectively, where $p_0 = \frac{d-K}{md}$, and for $1 \leq i \leq m$, $p_i = \frac{1}{m} - \frac{d-K}{m^2 d}$. The instances of U are placed in R_+^d as follows. u_0 is placed in the origin; that is, $u_0 = (0, 0, ..., 0)$. For each u_i ($1 \leq i \leq m$), its coordinate value on each dimension takes either 1 or 2 such that for $1 \leq j \leq d$, the jth coordinate value of u_i is 2 if $u_i \in S_j$ ($1 \leq j \leq d$), and 1 otherwise.

V has $(d+2)$ instances, v_0, v_1, ..., v_{d+1}, with the occurrence probabilities q_0, q_1, ... , q_{d+1}, respectively. Here, $q_0 = \frac{1}{m^3 d}$, $q_{d+1} = 1 - \frac{1}{m} - \frac{1}{m^3 d}$, and for $1 \leq i \leq d$, $q_i = \frac{1}{md}$. v_0 is placed at the origin and v_{d+1} is placed at $(2, 2, ..., 2)$. For $1 \leq i \leq d$, the ith coordinate value of v_i is 2 and the other coordinate values of v_i is 1.

Clearly, $\sum_{i=0}^{m} p_i = \sum_{i=0}^{d+1} q_i = 1$, $p_i > 0$ (for $0 \leq i \leq m$), and $q_i > 0$ (for $0 \leq i \leq d+1$). Note that for $1 \leq i \leq d$, $\mathcal{D}_i(V) = \{0, 1, 2\}$. For a (grid) point $x = (x_1, x_2, ..., x_d) \in \prod_{i=1}^{d} \mathcal{D}_i(V)$, the following can be immediately verified.

1. If $\exists i$ such that $x_i = 0$, then $cdf.U(x) > cdf.V(x)$; thus it cannot be a violation point regarding $U \prec_{sd} V$.

2. If each $x_i = 2$ for $1 \leq d$, then $cdf.U(x) = cdf.V(x) = 1$; thus it cannot be a violation point regarding $U \prec_{sd} V$.

3. If $x_{i_j} = 1$ for $1 \leq j \leq l$ ($l > 0$) and $x_j = 2$ otherwise, then the following two equalities hold where $S' = \cup_{j=1}^{l} S_{i_j}$.

$$cdf.V(x) = \frac{d-l}{md} + \frac{1}{m^3 d}. \tag{3.1}$$

$$cdf.U(x) = \frac{d-K}{md} + |S - S'|(\frac{1}{m} - \frac{d-K}{m^2 d}). \tag{3.2}$$

Regarding 3), since $S' \subseteq S$, if $S' \neq S$ (i.e. $\{S_{i_j} \mid 1 \leq j \leq l\}$ is not a cover of S), then $|S - S'| \geq 1$. Consequently, it can be immediately verified that $cdf.U(x) < cdf.V(x)$ if and only if $l \leq K$ and $S' = S$; that is, $C' = \{S_{i_j} \mid 1 \leq j \leq l\}$ is a cover of S with size K or less. Therefore, the minimum set cover problem has a positive answer if and only if in its corresponding instance, as constructed above, there is a grid point that is a violation point regarding $U \prec_{sd} V$; that is, if and only if U does not stochastically dominates V (by Theorem 2). □

3.3 Efficient Testing Algorithm

The NP-completeness implies that the exponential time complexity regarding d 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, $\mathcal{D}_i(V) = \{a_1, a_2, ..., a_l\}$, be sorted increasingly; that is, $a_i < a_{i+1}$ for $1 \leq i \leq l-1$. a_j $(1 < j < l)$ is a *switching* distinct value regarding U if there is at least one value in $\mathcal{D}_i(U)$ that is in $[a_j, a_{j+1}]$, and a_l (i.e., the maximum in $\mathcal{D}_i(V)$) is always a *switching* distinct value. 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 3.2(b) depicts $\prod_{i=1}^d D_i^s(V)$. Clearly, $D_i^s(V) \subseteq \mathcal{D}_i(V)$; consequently, we may reduce significantly the number of grid points for testing.

Theorem 4 *Assume that $U \neq V$. Then, V is not stochastically dominated by U if and only if there is a (grid) point $x \in \prod_{i=1}^d D_i^s(V)$ such that $cdf.V(x) - cdf.U(x) > 0$.*

Proof 2 *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 $R((0, ..., 0), x)$ are the same as those of U falling in $R((0, ..., 0), x')$; see such two pairs in Figure 3.2(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 2.*

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. Theorem 5 provides a sufficient condition to exclude any point in $R(x, y)$ to be a violation point regarding $U \prec_{sd} V$. Consequently, $R(x, y)$ can be discarded from the procedure of finding a violation point.

Theorem 5 *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 3 *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.*

Such an $R(x, y)$ in Theorem 5 is called *valid* regarding $U \prec_{sd} V$. To facilitate the observation in Theorem 5, our algorithm is partitioning-based, which iteratively divides rectangular regions into disjoint sub-rectangular regions. A job Q maintains a set of disjoint rectangular regions that cannot be discarded by Theorem 5 (i.e., not yet valid regarding $U \prec_{sd} V$). 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 $U \prec_{sd} V$. 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 sub-rectangular regions to be put into Q. Note that the corner points of a $R(x, y)$ are the points $(z_1, z_2, ..., z_d)$ such that for $1 \leq i \leq d$, z_i is x_i or y_i where $x = (x_1, ..., x_d)$ and $y = (y_1, y_2, ..., y_d)$. The algorithm terminates if a violation point is found or Q is $\emptyset$. Algorithm 1 below presents our partitioning based testing techniques. The algorithm outputs "true" if $U \prec_{sd} V$ and "false" otherwise.

Algorithm 1: **Verification(** $U \prec_{sd} V$ **)**

Input : objects U and V
Output: if $U \prec_{sd} V$ (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 ≺_sd 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
```

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 $U = (U_1, ..., U_d)$ and $V = (V_1, ..., V_d)$ where U_i and V_i ($1 \leq i \leq d$) are ith sub-variables of U and V, respectively. For each distinct value a_j in $D_i^s(V)$ ($1 \leq i \leq d$), Initial_Check ($U \prec_{sd} V$) calculates the total probability, denoted by $V.cdf(V_i \leq a_j)$, of the instances of V with its ith-coordinate value not great 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 $U \not\prec_{sd} V$ and Initial_Check ($U \prec_{sd} V$) returns *false*. The correctness of Initial_Check ($U \prec_{sd} V$) 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 $U \prec_{sd} V$ (i.e. the condition in Theorem 5) 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 $U \prec_{sd} V$. 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 2 *As depicted in Figure 3.3(a), assume that* V *has* 5 *switching distinct values*

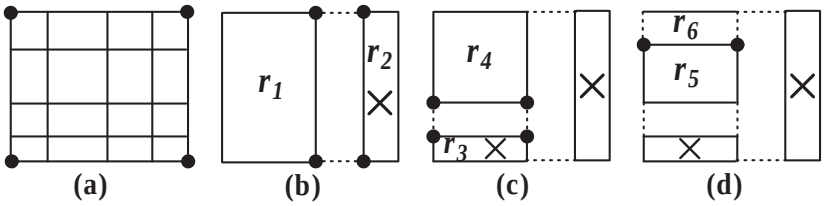


Figure 3.3: an example for splitting

in each dimension. Algorithm 1 splits V_{mbb} into r_1 and r_2 in Figure 3.3(b). Suppose that r_2 is discarded by Line 11. Algorithm 1 further splits r_1 into r_3 and r_4 in Figure 3.3(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_3 will be conducted on the line segment. In this example, at each iteration, solid points in Figure 3.3 indicate the new generated grid points.

Based on the correctness of Initial_Check ($U \prec_{sd} V$), Theorems 2, 4, and 5, 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 [24] 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 $\mathcal{D}_i(V) = \{a_1, ..., a_l\}$. 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 [24]. We also calculate $V.cdf(V_i \leq a_j)$ for each $a_j \in D_i^s(V)$ to conduct Initial_Check ($U \prec_{sd} V$). 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 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 $O((T(U_{artree}) + T(U_{artree})) \prod_{i=1}^{d} k_i)$ where $k_i = |D_i^s(V)|$. Clearly, checking if r is valid regarding $U \prec_{sd} V$ 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 $O((T(U_{artree}) + T(U_{artree})) \prod_{i=1}^{d} k_i)$. The following theorem is immediate based on the discussions above.

Theorem 6 *Algorithm 1 runs in time $O(dm \log m + m^d(T(U_{artree}) + T(U_{artree})))$ where m is the number of instances in V.*

The naive algorithm in Section 3.1 can also employ the window aggregate techniques in [24] 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 in Section 5 demonstrates that the naive algorithm is unpractical, while Algorithm 1 is very efficient in practice.

4 Stochastic Skyline Computation

We first present the index to be used, followed by our index-based framework, filtering techniques, a size estimation of stochastic skyline, and discussions.

4.1 Statistic R-Tree

As discussed in Section 3.3, 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, ... , u_m with the occurrence probabilities p_1, p_2, ... , p_m, respectively.

Definition 3 (*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 4 (*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 $\{E_1, E_2, ..., E_l\}$. 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 above statistic information, is called a statistic R-tree, denoted by sR-tree. Our algorithm for computing stochastic skyline is conducted against sR-tree of $\mathcal{U}$. To correctly use the verification algorithm (Algorithm 1), in this paper 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 $\mathcal{U}$.

4.2 Framework for stochastic skyline computation

It is immediate that for each point $x \in R_+^d$, if $U.cdf(x) \leq V.cdf(x)$ and $V.cdf(x) \leq W.cdf(x)$ then $U.cdf(x) \leq W.cdf(x)$. Therefore, $\prec_{sd}$ has the transitivity. Consequently, the standard filtering paradigm is applicable; that is, if $U \prec_{sd} V$ then V can be immediately removed since for any W, if $V \prec_{sd} W$ then $U \prec_{sd} W$ and W can be pruned by U.

Our index-based algorithm, Algorithm 2 below, adopts the branch and bound search paradigm [17]. It 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 with distances to the origin is not smaller than those of the early accessed objects. Consequently, based on Theorem 7 below 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.

Theorem 7 *For two U and V, if $dist(U_{min}) < dist(V_{min})$ then $V \not\prec_{sd} U$ where $dist(U_{min})$ and $dist(V_{min})$ denote the distances of U_{min} and V_{min} to the origin, respectively.*

Proof 4 *Immediately, $V_{min} \not\prec U_{min}$ and $U_{min} \neq U_{min}$. Thus, there must be an instance u in U such that $V_{min} \not\prec u$ and $V_{min} \neq u$. Therefore, $U.cdf(u) > 0$ and $V.cdf(u) = 0$. u is a violation point regarding $V \prec_{sd} U$.*

Algorithm 2: **stochastic skyline Computation**(sR)

Input : sR (sR-Tree for $\mathcal{U}$)
Output: R_{ssky} (stochastic skyline of $\mathcal{U}$)

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

Lines 2 and 16 push each child entry descriptions of the root or E, including its MBBs and the above 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_{ssky}, E) returns true if E is pruned by the current R_{ssky} using our filtering techniques in Section 4.3.

Line 8 performs the verification algorithm, Algorithm 1. 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 7. When $dist(U_{min}) = dist(V_{min})$, according to our algorithm U is not stochastically dominated by any objects accessed earlier, including those in the current R_{ssky}; nevertheless it is possible that V stochastically dominates U if V is not stochastically dominated by any object in the current R_{ssky}.

4.3 Filtering

A key in Algorithm 2 is to efficiently and effectively conduct PRUNE(R_{ssky}, E). The following two filtering techniques are developed to check if E can be pruned by U for each object U in R_{ssky} till R_{ssky} is exhausted or E is pruned.

1. MBB-based Pruning. The following pruning rule is immediate according to the definition of stochastic dominance 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; that is, E can be pruned.*

2. Statistic based Pruning. Our statistic based pruning technique uses the observation in Theorem 5 in combining with the Cantelli's Inequality [16].

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.

Let $U_{max} = (a_1, a_2, ..., a_d)$ and $E_{min} = (b_1, b_2, ..., b_d)$. Below we present an effective pruning rule for two cases to prune E by U when E cannot be pruned by Pruning Rule 1, $U_{min} \preceq E_{min}$, and $U_{max} \prec E_{max}$. **Case 1**: $\exists i_1 \& i_2$ such that $a_{i_1} > b_{i_1}$, $a_{i_2} > b_{i_2}$, and $a_i \leq b_i$ if $i \neq i_1$ and $i \neq i_2$. **Case 2**: $\exists i$ such that $a_i > b_i$ and $a_j \leq b_j$ if $j \neq i$. Figures 4.1(a) and 4.1(b) depicts these two cases with E being a data entry of the sR-tree - the MBB of an object V.

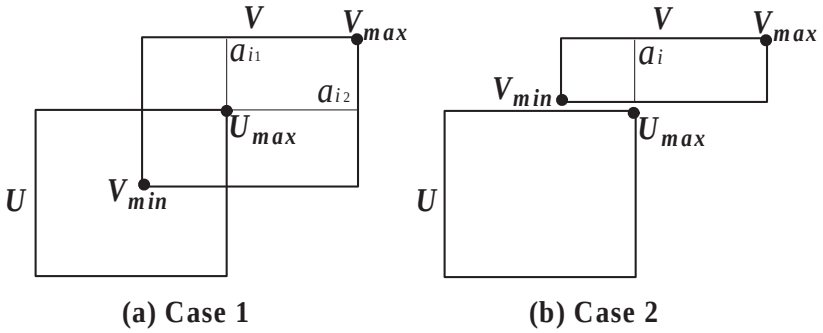


Figure 4.1: Statistic Pruning

We firstly assume that E is a data entry; that is, E is V_{mbb}. For the case 1 above, we use a_{i_1} and a_{i_2} to divide V_{mbb} into 3 rectangular regions: 1) $r1$: the points $(x_1, x_2, ..., x_d) \in V_{mbb}$ with the constraint that on the i_1th dimension, $x_{i_1} \leq a_{i_1}$; 2) $r2$: the points $(x_1, x_2, ..., x_d) \in V_{mbb}$ with the constraint that on the i_2th dimension, $x_{i_2} \leq a_{i_2}$; and 3): $r3$: the points $(x_1, x_2, ..., x_d) \in V_{mbb}$ such that for $1 \leq i \leq d$, $a_i \leq x_i \leq b_i$. Figure 4.1(a) shows an example. Note that for the ease of statistic estimation below, $r1$ and $r2$ share a common area. It can be immediately verified that $U.cdf(r3_{min}) = U.cdf(U_{max}) = 1 = V.cdf(V_{max}) = V.cdf(r3_{max})$; thus, $r3$ is always valid regarding $U \prec_{sd} V$. Note that $V.cdf(r1_{max}) = V.cdf(V_{i_1} \leq a_{i_1})$ and $V.cdf(r2_{max}) = V.cdf(V_{i_1} \leq a_{i_2})$, where U_i and V_i are the ith sub-variables of U and V (i.e. $U = (U_1, ..., U_d)$ and $V = (V_1, ..., V_d)$), respectively. Therefore, based on Theorems 5 and 2, if $r1$ and $r2$ are valid regarding $U \prec_{sd} V$ then $U \prec_{sd} V$ since $r1$,

$r2$, and $r3$ covers V_{mbb}. Consider that $V_{min} = r1_{min} = r2_{min}$. Immediately, $r1$ and $r2$ are valid regarding $U \prec_{sd} V$ if and only if

$$U.cdf(V_{min}) \geq \max\{V.cdf(V_{i_1} \leq a_{i_1}), V.cdf(V_{i_2} \leq a_{i_2})\} \tag{4.1}$$

For the case 2 above, a_i divides V_{mbb} into 2 rectangular regions: 1) $r1$: the points $(x_1, x_2, ..., x_d) \in V_{mbb}$ such that $x_i \leq a_i$; and 2) $r2$: the points $(x_1, x_2, ..., x_d) \in V_{mbb}$ such that $x_i \geq a_i$. Figure 4.1(b) shows an example. It can be immediately verified that $U.cdf(r2_{min}) = U.cdf(U_i \leq a_i) = V.cdf(r2_{max}) = 1$; thus $r2$ is always valid regarding $U \prec_{sd} V$. Similarly, it is immediate that $r1$ is valid regarding $U \prec_{sd} V$ if and only if

$$U.cdf(V_{min}) \geq V.cdf(V_i \leq a_i) \tag{4.2}$$

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* [16] 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 8 (Cantelli's Inequality [16]) *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 8 extends the original Cantelli's Inequality [16] 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 9 *Assume that $0 \leq b \leq \mu(t)$. Then, $Prob(t \leq b) \leq \delta(\mu(t) - b, \sigma(t))$.*

Proof 5 *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.*

Regarding case 1, $\Delta_1(E, U) = \max\{\delta(\mu_{i_1}(E) - a_{i_1}, \sigma^2_{i_1}(E)),\ \delta(\mu_{i_2}(E) - a_{i_2}, \sigma^2_{i_2}(E))\}$ when $a_{i_1} \leq \mu_{i_1}(E)$ and $a_{i_2} \leq \mu_{i_2}(E)$; and $\Delta_1(E, U) = \infty$ otherwise. Regarding case 2, $\Delta_2(E, U) = \delta(\mu_i(E) - a_i, \sigma^2_i(E))$ if $a_i \leq \mu_i(E)$; and $\Delta_2(E, U) = \infty$ otherwise.

Pruning Rule 2 *Suppose that $U_{min} \preceq E_{min}$ and $U_{max} \prec E_{max}$. If the following conditions hold, then every object in the entry E of the global R-tree is stochastically dominated by U; that is E can be pruned by U.*

1. *When U_{max} and E_{min} fall in case 1, $U.cdf(E_{min}) \geq \Delta_1(E, U)$.*
2. *When U_{max} and E_{min} fall in case 2, $U.cdf(E_{min}) \geq \Delta_2(E, U)$.*

Proof 6 *Note that each uncertain object V can also be regarded as a random variable $(V_1, V_2, ..., V_d)$. Immediately, each sub-variable V_i of V can be regarded as a random variable in R^1 with the mean $\mu_i(V)$ and the variance $\sigma^2_i(V)$. From Theorem 5, the inequalities in (4.1) and (4.2), the assumptions in Section 4.1 that no objects indexed by sR-tree are equal, and Theorem 9, this Pruning Rule immediately holds if E is a data entry V_{mbb}; that is, $U \prec_{sd} V$.*

Secondly, suppose that E is an intermediate entry of the global sR-tree. If U_{max} and E_{min} fall in case 1, then an object V contained by E either meets the condition in Pruning Rule 1, or U_{max} and V_{min} fall in case 1 or case 2. Consider that $\delta(\mu - b, \sigma)$ is increasing and decreasing regarding σ and μ, respectively, when $\mu - b \geq 0$ and $\sigma \geq 0$, and $\sigma_i^2(E)$ and $\mu_i(E)$ are chosen as the maximum and minimum values among the objects contained. Immediately, $U \prec_{sd} V$ since the conditions in 1) and 2) hold between U and V, respectively, regarding case 1 and case 2. Similarly, we can show that $U \prec_{sd} V$ for each object V contained by E if U_{max} and E_{min} fall in case 2; in this situation, case 1 does not occur.

4.4 Analysis of Algorithm 2

Prune(R_{ssky}, E) in Algorithm 2 is conducted as follows that 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 $U \in R_{ssky}$.

Note that a more general version of Pruning Rule 2 may be developed following a similar idea. Due to space limits, it is omitted in the paper since the gain by doing this in practice is very limited in our initial experiments.

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

Access Order of R_{ssky}. In Algorithm 2, 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.

4.5 Discussions

Size Estimation. It tends to be quite complicated to estimate the size of stochastic skyline 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 stochastic skyline objects is bounded by $\ln^d(n)/(d+1)!$ - the expected size of conventional skyline in $(d+1)$ dimension space [7], where n is the number of objects in $\mathcal{U}$. The empirical study in Section 5 also shows that the size of stochastic skyline objects in d-dimension space is almost between those of conventional skyline in d-dimensional space and $(d+1)$-dimensional spaces.

Theorem 10 *Given a set of objects $\mathcal{U}$, assume that MBBs of all objects are hyper-cubes. We assume the pdfs 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 size of $ssky(\mathcal{U})$ is bounded by $\ln^d(n)/(d+1)!$.*

Proof 7 *$\forall U \in \mathcal{U}$, 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 stochastic skyline 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* $\ln^d(n)/(d+1)!$ *[7].*

Continuous Cases. In the paper, we focus on discrete cases of probability distributions. For continuous cases, we can discrete a continues 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.

Other Statistic Orders. There are two other popular stochastic orders defined in the literature [21], 1) *upper orthant order*, and 2) *usual stochastic order*. We can also define stochastic skyline against these two orders, respectively. Note that the upper orthant order is "symmetric" to the lower orthant order with the preference on larger values; thus the techniques developed in the paper can be immediately modified to the stochastic skyline regarding the upper orthant order.

The usual stochastic order is defined below [21].

Definition 5 (Usual Stochastic Order) *Given objects* U *and* V*,* U dominate*s* V*, denoted by* $U \prec_{st} V$ *if for any upper space* S*,* $U.cdf(S) \leq V.cdf(S)$*.*

Note that $S \subseteq R^d$ is an *upper space* if for any $x, y \in R^d$, $x \preceq y$ and $x \in S$, we have $y \in S$. Similar results to Theorem 1 presented in page 266 of [21] imply that the stochastic skyline regarding the usual stochastic order can provide the minimum sets of candidate for the optimal solutions (with maximum expected values) regarding any forms of decreasing functions; that is, stochastic skyline excludes objects that are not preferred by any decreasing functions.

Note that the number of skyline objects regarding the usual stochastic order is usually larger than the number of skyline objects regarding the lower orthant order, since the lower orthant order may be regarded as a special case of usual stochastic order. However, the size estimation result in Theorem 10 also holds. New techniques for computing the stochastic skyline regarding the usual stochastic order need to be developed. For instance, the verification is a much harder problem.

5 empirical study

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

- **ssky**: Algorithm 2 proposed in Section 4 to compute stochastic skyline.
- **ssky-NF** : **ssky** without the filtering techniques in Section 4.3 but with the non-naive verification algorithm (Algorithm 1) in Section 3.3.
- **ssky-NV**: **ssky** with the two filtering techniques in Section 4.3 and the naive verification algorithm in Section 3.1.

5.1 Experiment Setup

All algorithms proposed in the paper are implemented in standard C++ with STL library support and compiled with GNU GCC. Experiments are conducted on a PC with

Intel Xeon 2.4GHz 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 2048 bytes. The instances of an object are organized by a main-memory based aggregate R-tree with fan-out 8 and we load in the whole aggregate R-tree of V if V cannot be pruned by the filtering techniques.

We use both real and synthetic data sets in our evaluation process.

Real dataset is extracted from NBA players' game-by-game statistics (http://www.nba.com), containing 339,721 records of 1,313 players. Each player is treated as an uncertain object where the statistics of a player per game is treated as an instance. For one player, all instances are assumed to take the same probability to appear. In our experiment, we use three attributes, points, assistances, and rebounds in an instance. 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 datasets are generated using methodologies in [4] with respect to the following parameters. The centers of objects (objects' MBBs) follow either *anti-correlated* (*anti* for short), *correlated* (*corr*) or *independent* (*inde*) distribution. We use **anti** as the default distribution for objects' centers. Data domain in each dimension is $[0, 1]$. The MBBs of the objects are hype-cube with average edge length h varying from 0.02 to 0.1 with the default value 0.04. The average number of instances m in each object varies from 200 to 1000 with the default value 400. Locations of instances of an object follow one of the 4 distributions, *uniform* (*unif* for short), *zipf*, *constrained normal* (*norm*) or a mixture of the previous three *mix*. In *unif*, instances are distributed uniformly inside an MBB with the *same occurrence probability*. In *zipf*, firstly an instance u of U is randomly generated and the distances from all other instances to u follow a zipf distribution with $z = 0.5$. In *norm*, instances follow the normal distribution within the MBR of the object with standard deviation $\sigma = 0.4 \times h$. In *mix*, previous three distributions are mixed each with a portion of $\frac{1}{3}$. We use **mix** as the default distribution for instances. Note that in a synthetic dataset, the instances in every object have the same probability value.

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 5.1: Parameters

We also study the impact of other key parameters. 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 5.1 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 below, these parameters use default values unless otherwise specified.*

p	#psky	#hit
0.5	0	0
0.05	83	76
0.005	316	122
5×10^{-10}	837	124

(a) NBA(#ssky: 127)

p	#psky	#hit
0.8	12	12
0.08	137	94
0.008	310	124
3×10^{-10}	1215	147

(b) 3d (#ssky: 148)

Table 5.2: pskyline vs stochastic skyline

5.2 Size of Stochastic Skyline

Table 5.2 shows the result sizes (#psky) of *probabilistic skyline* and the number (#hit) of stochastic skyline objects contained by the corresponding *probabilistic skyline* regarding different probability thresholds. The real dataset NBA data is employed, as well as a 3 dimension synthetic dataset with instance locations following *unif* and MBB centers following *inde* (other parameters are default values). The experiment result shows that some stochastic skyline objects may have very small skyline probabilities and some non-stochastic skyline objects may have large skyline probabilities. Consequently, it shows that we may have to use very small probability threshold (p) to generate the *probabilistic skyline* with large size to cover all stochastic skyline objects.

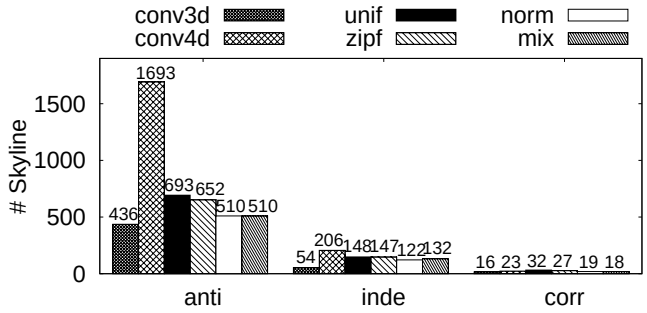


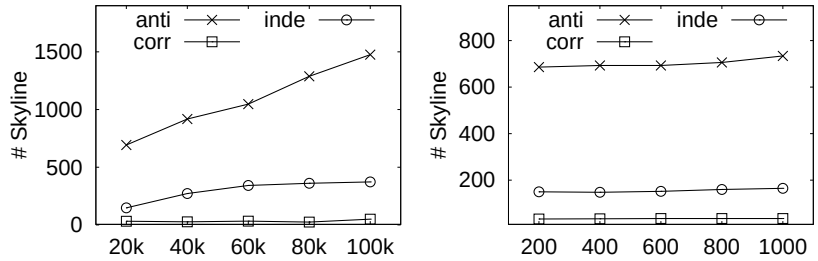
Figure 5.1: Skyline Size vs Different Distributions

Figure 5.1 evaluates the impacts of different distributions of instances locations and object MBB centers, respectively. We use *conv* to denote the size of (conventional) skyline against the set of MBB centers. The x-coordinate in Figure 5.1 gives different distributions of MBB centers; regarding each distribution of MBB centers, we record the number of skyline objects over one distribution of instance locations. We also record the size of (conventional) skyline against the set of MBB centers in 4-dimensional space to evaluate Theorem 10 in practice since the uncertain data is in 3-dimensional space.

Figure 5.2 reports the skyline size regarding the number of objects and average number of instances per object. As expected, the skyline size grows with the number of objects but is not very sensitive to the number of instances per object.

5.3 Evaluating Efficiency

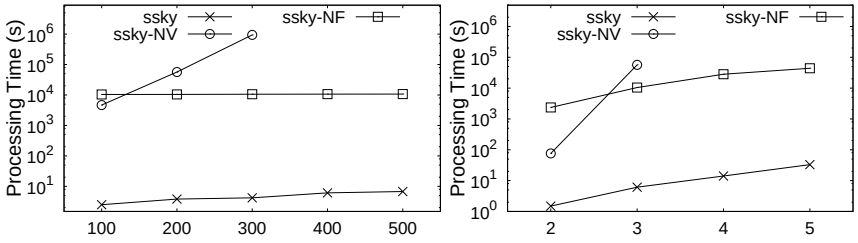
We first evaluate the efficiency of **ssky** to compute stochastic skyline. We have done the comparison of **ssky**, **ssky**-NF and **ssky**-NV to evaluate the overall performance of



(a) #Objects (b) #Instances

Figure 5.2: Skyline Sizes vs Data Sizes

filtering techniques and verification techniques. Since the naive verification algorithm is very inefficient comparing with the non-naive verification algorithm Algorithm 1, we use small data size in the experiment where only $10k$ objects are involved and the average number of instances in Figure 5.3(b) is 200. As depicted in Figures 5.3(a)-(b), with naive verification approach, the performance of **ssky**-NV drops quickly with the increase of instance number and d due to the time complexity $O(m^d)$. Some values regarding **ssky**-NV are missing because under these settings, we can not get result after 5 days running. Moreover, without any filtering techniques, the performance of our algorithm, i.e. **ssky**-NF algorithm, is very poor due to the large number of IO accesses and verifications. **ssky**-NF is orders of magnitude slower than **ssky**.



(a) #Instances (b) d

Figure 5.3: Comparison with Naive Techniques

Therefore, in the rest of experiments we no-longer evaluate **ssky**-NF (without filtering techniques) and **ssky**-NV (with the naive verification algorithm). We focus on evaluating the two filtering techniques. Particularly, we evaluate the performance of **ssky** and **ssky-F2** where **ssky-F2** stands for that in Algorithm 2, only Pruning Rule 1 is used.

Figure 5.4 reports the evaluation of **ssky** and **ssky-F2** against different distributions of instance locations and NBA data. Both algorithms are quite efficient. It also shows that **ssky** always significantly outperforms **ssky-F2**; that is, the Pruning Rule 2 is very effective in practice. Based on the experiment results in Figures 5.3(a)-(b) which demonstrate that **ssky**-NF (i.e. without the two filtering techniques) is orders of of magnitude slower than **ssky**, this experiment implies that the 1st filtering technique is also very effective.

In Figure 5.5, we evaluate the scalability of our algorithms against different dataset sizes, number of instances, MBB sizes, and dimensionality. Figure 5.5(a) and Fig-

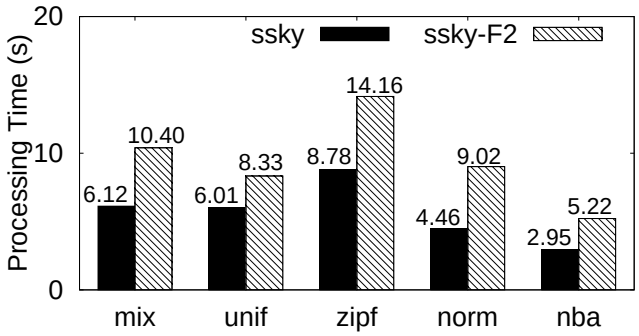


Figure 5.4: Processing Time w.r.t Diff. Distributions

ure 5.5(b) show that the performance of **ssky** and **ssky-F2** degrades "linearly" with the growth of dataset size, number of instances, while in Figure 5.5(c) and Figure 5.5(d) the performance drops more significantly with the growth of MBB edge size and dimensionality due to the increase of the number of stochastic skyline objects. Nevertheless, the gain of statistics pruning rules becomes more significant when dimensionality is high and MBB edge size is large.

Figure 5.6 reports the number of objects accessed regarding dataset sizes and the dimensionality. Both algorithms, **ssky** and **ssky-F2**, are I/O efficient because a large number of objects are eliminated by Pruning Rules 1 and/or 2. For instance, Figure 5.6(a) shows that only 11.2% and 5.6% of objects are loaded into main memory by algorithms **ssky-F2** and **ssky**, respectively, when the number of objects is $100k$. Figure 5.6 also shows that Pruning Rule 2 can further improve the I/O efficiency. As expected, the number of object (I/O) accesses increases with the increase of dataset size and dimensionality.

Table 5.3 gives a breakdown information of the filtering time and verification (in seconds). The results are reported in where we vary the number of objects from $20k$ to $100k$. It shows the filtering costs are much smaller than the costs of verification.

	20k	40k	60k	80k	100k
Filtering	0.25	0.45	0.58	0.80	2.50
Verification	5.96	10.10	17.45	20.22	31.21

Table 5.3: Filtering and Verification time(s)

6 Related Work

While the paper is the first work to model and efficiently compute stochastic skyline, below we give a brief overview of skyline computation over conventional and uncertain data, respectively.

Conventional Skyline Computation. Börzsönyi *et al* [4] firstly study the problem of computing skylines over large datasets. They develop *block-nested-loop* (BNL) and *divide-and-conquer* (D & C) based techniques for skyline computation. The *Sort Filter Skyline* (SFS) algorithm [8] aims to improve BNL by sorting a dataset first. An optimized version of SFS, named *linear elimination sort for skyline* (LESS) is later

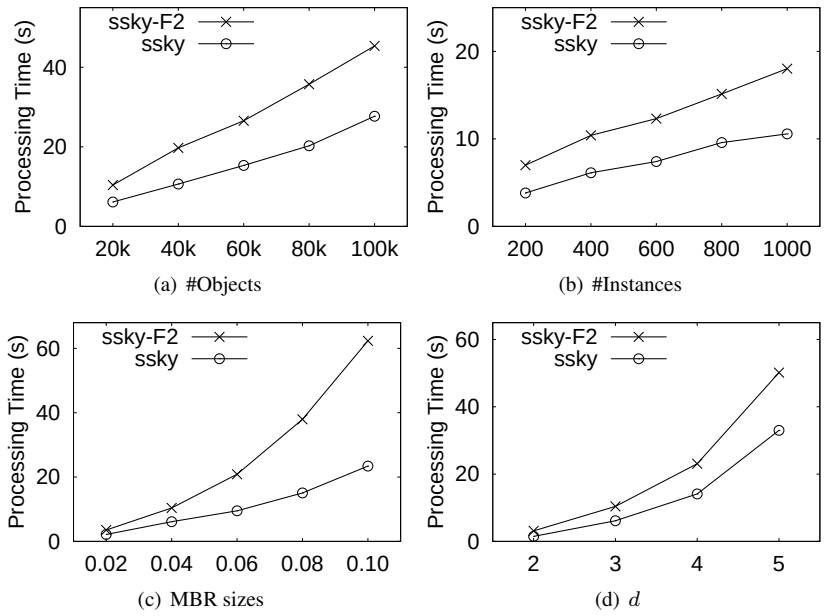


Figure 5.5: Processing Time w.r.t Different Parameters

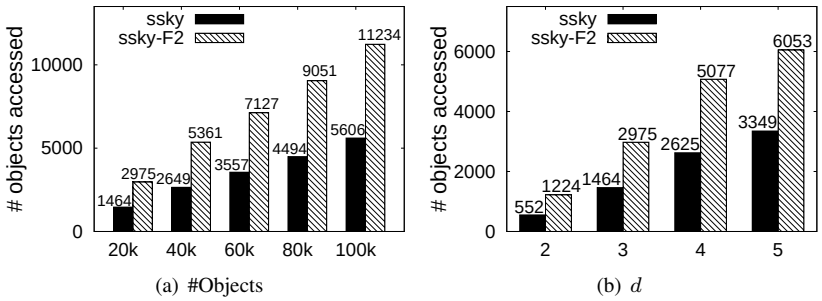


Figure 5.6: I/O costs w.r.t Different Parameters

proposed in [10]. *Sort and limit skyline algorithm* (SaLSa) [3] aims to improve SFS and LESS by avoiding scanning the complete set of sorted objects.

The first index based techniques are proposed by Tan *et al* [23] where two progressive techniques, *Bitmap* and *Index* based on bitmap and B-tree structures, are developed. Various index based techniques are also developed (e.g. [12, 13, 17]). Very recently, an effective dynamic indexing technique is proposed in [25] to index the current skyline.

Variations of skyline computation have also been extensively explored; for example, skylines for partially-ordered value domains[6] and skyline cubes [19].

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 [1, 5]). Sarma *et al* [20] purpose to model queries over uncertain data by *possible world semantics*.

Probabilistic skyline on uncertain data is first tackled by Pei *et al* [18] where skyline objects are retrieved based on skyline probabilities. Efficient techniques are pro-

posed following the bounding-pruning-refining framework. Lian *et al* [15] combine reverse skyline with uncertain semantics and study the probabilistic reverse skyline problem in both monochromatic and bichromatic fashion. Atallah and Qi [2] develop sub-quadratic algorithms to compute skyline probabilities for every object. Zhang *et al* [26] tackle the problem of efficiently on-line computing probabilistic skyline over sliding windows.

7 conclusion

In this paper, we propose a novel stochastic skyline model based on the *stochastic orders* which strictly captures the preference of users and guarantees to provide the minimum set of candidates to the optimal solutions over a broad and popular family of functions. We develop efficient stochastic skyline computation algorithm on large set of objects based on novel filtering and verification techniques. Comprehensive experiments are conducted on both real and synthetic data to demonstrate the efficiency of our techniques. As a possible future work, we will investigate the problem against correlated uncertain data, as well as the stochastic skyline computation over other stochastic orders.

Bibliography

[1] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha Nabar, Tomoe Sugihara, and Jennifer Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, 2006.

[2] Mikhail J. Atallah and Yinian Qi. Computing all skyline probabilities for uncertain data. In *PODS*, 2009.

[3] I. Bartolini, P. Ciaccia, and M. Patella. Efficient sort-based skyline evaluation. In *ACM TODS*, 2008.

[4] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *ICDE*, 2001.

[5] Jihad Boulos, Nilesh Dalvi, Bhushan Mandhani, Shobhit Mathur, Chris Re, and Dan Suciu. MYSTIQ: A system for finding more answers by using probabilities. In *SIGMOD*, 2005.

[6] Chee-Yong Chan, Pin-Kwang Eng, and Kian-Lee Tan. Stratified computation of skylines with paritally ordered domains. In *SIGMOD*, 2005.

[7] Surajit Chaudhuri, Nilesh N. Dalvi, and Raghav Kaushik. Robust cardinality and cost estimation for skyline operator. In *ICDE*, 2006.

[8] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *ICDE*, 2003.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guider to the Theory of NP-Completeness*. 1990.

[10] P. Godfrey, R. Shipley, and J. Gryz. Maximal vector computation in large data sets. In *VLDB*, 2005.

[11] Masaaki Kijima and M. Ohnishi. Stochastic orders and their applications in financial optimization. *Mathematical Methods of Operations Research*, 50(2), 1999.

[12] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, 2002.

[13] K. C. K. Lee, B. Zheng, H. Li, and W. C. Lee. Approaching the skyline in z order. In *VLDB*, 2007.

[14] Haim Levy. Stochastic dominance and exptected utility: survey and analysis. *Management Science*, 38(4), 1992.

[15] Xiang Lian and Lei Chen. Monochromatic and bichromatic reverse skyline search over uncertain databases. In *SIGMOD 2008*.

[16] Ronald Meester. *A Natural Introduction to Probability Theory*. 2004.

[17] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal progressive algorithm for skyline queries. In *SIGMOD*, 2003.

[18] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. Probabilistic skylines on uncertain data. In *VLDB*, 2007.

[19] Jian Pei, Yidong Yuan, Xuemin Lin, Wen Jin, Martin Ester, Qing Liu, Wei Wang, Yufei Tao, Jeffrey Xu Yu, and Qing Zhang. Towards multidimensional subspace skyline analysis. *ACM Trans. Database Syst.*, 31(4), 2006.

[20] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. In *ICDE*, 2005.

[21] Moshe Shaked and J. George Shanthikumar. *Stochastic Orders and Their Applications*. Academic Press, 2007.

[22] Ralph E. Steuer. *Multi Criteria Optimization: Theory, Computation, and Application*. John Willy and Sons, 1995.

[23] Kian-Lee Tan, Pin-Kwang Eng, and Beng Chin Ooi. Efficient progressive skyline computation. In *VLDB*, 2001.

[24] Yufei Tao and Dimitris Papadias. Range aggregate processing in spatial databases. *TKDE*, pages 1555–1570, 2004.

[25] Shiming Zhang, Nikos Mamoulis, and David W. Cheung. Scalable skyline computation using object-based space partitioning. In *SIGMOD Conference*, 2009.

[26] Wenjie Zhang, Xuemin Lin, Ying Zhang, Wei Wang, and Jeffrey Xu Yu. Probabilistic skyline operator over sliding windows. In *ICDE*, 2009.