Skyline Operator on Anti-correlated Distributions

Haichuan Shang[†] Masaru Kitsuregawa[†‡]
[†]Institute of Industrial Science, The University of Tokyo, Japan
[‡]National Institute of Informatics, Japan
{shang, kitsure}@tkl.iis.u-tokyo.ac.jp

ABSTRACT

Finding the skyline in a multi-dimensional space is relevant to a wide range of applications. The skyline operator over a set of d-dimensional points selects the points that are not dominated by any other point on all dimensions. Therefore, it provides a minimal set of candidates for the users to make their personal trade-off among all optimal solutions.

The existing algorithms establish both the worst case complexity by discarding distributions and the average case complexity by assuming dimensional independence. However, the data in the real world is more likely to be anti-correlated. The cardinality and complexity analysis on dimensionally independent data is meaningless when dealing with anti-correlated data. Furthermore, the performance of the existing algorithms becomes impractical on anti-correlated data.

In this paper, we establish a cardinality model for anti-correlated distributions. We propose an accurate polynomial estimation for the expected value of the skyline cardinality. Because the high skyline cardinality downgrades the performance of most existing algorithms on anti-correlated data, we further develop a determination and elimination framework which extends the well-adopted elimination strategy. It achieves remarkable effectiveness and efficiency. The comprehensive experiments on both real datasets and benchmark synthetic datasets demonstrate that our approach significantly outperforms the state-of-the-art algorithms under a wide range of settings.

1. INTRODUCTION

In a d-dimensional space, a point x is dominated by another point y if the coordinate of y is smaller than that of x on one dimension, and smaller than or equal to that of x on all other dimensions. Given a set of points $p_1, p_2, ..., p_n$ in a d-dimensional space, the skyline operator returns all the points p_i such that p_i is not dominated by any other point p_j. A classic example in literature is shown in Figure 1. This example describes a database containing a set of hotels. For each hotel the database stores its distance to the beach (x axis) and its price (y axis). According to the dominance definition, a hotel dominates another if it is cheaper and closer to the beach. In this example, the skyline consists of four points a, b, c, and d.

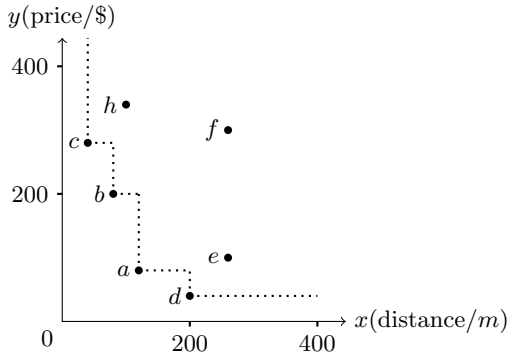


Figure 1: A skyline example

The existing studies explore the skyline problem by either discarding data distributions or assuming dimensional independence. Unfortunately, the data in real world applications usually does not satisfy these two conditions. In common experiences, the hotels with decent quality and attractive location are always expensive. In statistics, this phenomenon is named anti-correlation. For a point in a d-dimensional space, the anti-correlation means a relationship in which the value in one dimension increases as the values in the other dimensions decrease. The anti-correlation significantly limits the practical usage of the existing algorithms and yields the demand of effective mathematical models and efficient algorithms on anti-correlated data.

In order to develop the skyline operator on anti-correlated data, the first major challenge is to model anti-correlated distributions. Although there is much common sense about anti-correlations, none of the existing work formally models anti-correlated distributions. It is challenging to model an anti-correlated distribution with respect to how serious the anti-correlation happens. The second difficult yet important issue is to compute and estimate the skyline cardinality. The estimation should be accurate and efficient. Finally, it is challenging to develop efficient algorithms on anti-correlated distributions. Computing skyline on anti-correlated data is much more time-consuming than that on dimensionally independent data. The performance of the existing work is usually impractical on anti-correlated distributions, because

Proceedings of the VLDB Endowment, Vol. 6, No. 9

these algorithms are sensitive to skyline cardinalities which are relatively large on anti-correlated data.

We address the above issues with focuses on comprehensively modeling anti-correlation, effectively estimating skyline cardinality, and efficiently processing skyline queries. Our contributions can be summarized as follows.

- We propose a general model for the anti-correlated distributions and analyze the lower and upper bounds of the expected value of the skyline cardinality. Our study shows that:

$$\sum_{k=1}^{d}(-1)^{k-1}\binom{d-1}{k-1}n\frac{\Gamma(\frac{k}{d})\Gamma(n)}{\Gamma(n+\frac{k}{d})} \leq \hat{S}_{d,n} \leq n$$

  for $d \geq 2$ where $\Gamma(n) = \int_0^\infty e^{-t}t^n dt$ and $\hat{S}_{d,n}$ is the expected value of the skyline cardinality with n points and d dimensions.
- We propose an effective polynomial estimation of the lower bound of the expected value of the skyline cardinality. That is:

$$\sum_{k=1}^{d}(-1)^{k-1}\binom{d-1}{k-1}\Gamma(\frac{k}{d})n^{1-\frac{k}{d}} \leq \hat{S}_{d,n} \leq n$$

  for $d \geq 2$ where $\Gamma(n) = \int_0^\infty e^{-t}t^n dt$ and $\hat{S}_{d,n}$ is the expected value of the skyline cardinality with n points and d dimensions. This estimation can be further abbreviated as:

$$O(n^{\frac{d-1}{d}}) \leq O(\hat{S}_{d,n}) \leq O(n)$$

- We develop efficient algorithms to compute skyline on anti-correlated distributions. In order to solve the high skyline cardinality challenge, our proposed techniques not only effectively eliminate non-promising points, but also efficiently determine skyline points. Taking into account the elimination strategy applied by the existing work, our approach extends this strategy to build a *determination and elimination* framework.
- Besides the theoretical results, we also conduct a comprehensive experimental evaluation demonstrating that our algorithm outperforms the state-of-the-art algorithms under a wide range of settings. The performance gap is up to two orders of magnitude.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 presents the preliminaries and the problem definition. Section 4 analyzes the skyline cardinality on anti-correlated distributions. Section 5 introduces our proposed algorithms. Section 6 reports the experimental results and analyses. The conclusion is given in Section 7.

2. RELATED WORK

In this study, we focus on the skyline algorithms that can be directly implemented on raw data without preprocessing. We summarize the existing work into two categories: worst case complexity category and elimination category. The algorithms in the worst case complexity category study the worst case complexity on arbitrary data distributions. In contrast, most of the algorithms in the elimination category assume the data is either *independent* (all dimensions are independent) or *independent-and-uniform* (all dimensions are independent and the values follow a uniform distribution in each dimension). Under such an assumption, the algorithms explore the expected value of the complexity.

There are many other studies (e.g. [15, 18, 19]) which concern preprocessing the data into specialized data structures to facilitate the retrieval of the skyline.

2.1 Worst-case Complexity Category

In this category, the algorithms focus on the worst-case complexity.

Kung et al. [16] prove the complexity lower bound as $\Omega(\lceil \log n! \rceil)$ for $d \geq 2$ which is approximately $\Omega(n \log n)$. They also propose $O(n \log n)$ algorithm for $d = 2, 3$. Together with the lower bound, they establish the complexity $\Theta(n \log n)$ which is optimal for $d = 2, 3$. For any $d > 3$, Kung et al. [16] further propose a general divide-and-conquer algorithm in $O(n \log^{d-2} n)$ complexity. Based on the similar idea of divide-and-conquer, Bentley [3] develops alternative algorithm achieving the same complexity. In a recent study, Sheng et al. [21] present that the $d = 2$ solution of Kung et al. [16] can be easily adapted as an external algorithm which is still optimal in $\Theta((N/B)\log_{M/B}(N/B))$ I/Os.

Borzsonyi et al. [7] propose an external version of divide-and-conquer. Their method divides the points into m-partitions such that the skyline of each partition can be computed in memory. The final answer is produced by merging the skylines pairwise. However the performance bound of this method remains unknown.

Sheng et al. [21] discover an external algorithm terminating in $O((N/B)\log^{d-2}_{M/B}(N/B))$ I/Os. The algorithm solves the obstacle of computing the skyline using an external divide-and-conquer approach. That is a distribution-sweep algorithm for solving the skyline merge problem.

2.2 Elimination Category

However, the algorithms in the worst-case complexity category only deal with the worst-case complexity. It is often more interesting and practical to consider the average-case complexity. In the elimination category, the points are drawn from a probability distribution. For most algorithms, it is a distribution with dimensional independence.

Bentley et al. [4] divide the points by a virtual point which is ranked as $n(\ln n/n)^{1/d}$ (i.e. $n(1-(\ln n/n)^{1/d})$ in maxima problem) in each dimension. The probability that no point dominates the virtual point is bounded by $1/n$. If this event occurs, they adopt the worst-case algorithm [16] by Kung et al. Otherwise, the points which are dominated by the virtual point can be safely eliminated. The expected number of the remaining points is bounded by $1 + dn(\ln n/n)^{1/d}$ or $1 + d(n^{1-1/d}\ln^{1/d} n)$ and can be partitioned into d hyperrectangles with N^P property of Bentley et al. [6]. This subproblem can be computed in linear expected time by the algorithm in [6]. As a result, the algorithm achieves linear expected time under the dimensionally independent assumption.

Borzsonyi et al. [7] present block-nested-loop (BNL). This algorithm keeps a window of incomparable tuples in main memory. The input file is scanned. Each point in the input file is compared with points in the window. If it is dominated by any of them, it is eliminated. If it dominates any window points, the algorithm eliminates these window points and inserts the scanned point into the window. If neither of the above two situations happen, the scanned point is incomparable with all the window points. If there is enough room in

the window, the scanned point is inserted into the window. Otherwise, the algorithm stores it in a temporary file. After the first iteration, the window points are incomparable with the points in the temporary file and can be output as the result. Then, the next iteration is conducted on the temporary file. This algorithm uses heuristic elimination and works well if the skyline is small and the window points can eliminate most points.

Chomicki et al. [9] develop *sort filter skyline* (SFS). It sorts the points descendingly by the volume which are dominated by the points. The advantage is that the points in the beginning of the sorted list have a higher chance to dominate many other points than the points in the end of the sorted list.

Godfrey et al. [12] describe *linear elimination sort for skyline* (LESS) which integrates the advantages of BNL and SFS. Instead of using a standard external sorting on the data, LESS maintains an elimination-filter window (similar to BNL) in the external sorting procedure. The elimination-filter window keeps the copies of the points with best volume scores (similar to SFS) and eliminates the points which are dominated by the window points. After the sorting procedure, only the points which are not dominated by the window points remain. The rest of the algorithm is the same as SFS. This algorithm avoids the high complexity external sorting of SFS. It achieves linear expected-time when the dimensions are independent and the values follow a uniform distribution in each dimension.

Bartolini et al. [1] exploit the idea of sorting points according to their minimum coordinate value among all dimensions to effectively limit the number of tuples to be read and compared. Similar to SFS, the proposed algorithm SaLSa supports the systems on which skyline queries are executed as a client system.

Zhang et al. [22] tackle the problem of CPU-intensive skyline computing in high dimensional spaces. The proposed indexing method aims to improve the performance of sort-based skyline algorithms. Instead of maintaining the skyline points in an array, This approach partitions the skyline points into 2^d hypercubes based on a given skyline point and maintains $2^d - 2$ of the partitions as the children of the given point in tree structure. Recursively using this schema, this approach organizes the current skyline points in a search tree. Therefore some of the comparisons are avoided by using the search tree. However, the cost analysis of this approach is under uniform-and-independence assumption and this approach still falls into the linear expected-time category as the others.

Lee et al. [17] propose an alternative implementation of [22]. Both approaches rely on quadtrees in which each tree node uses a data item to split the underlying space. The algorithm in [17] exploits the idea of selecting the optimal skyline point to partition the data space. This idea is alternative to the OSPSOnSortingFirst algorithm in [22] which adopts the state-of-the-art sorting function [9] to find the partitioning point.

Sarma et al. [20] propose a skyline cardinality sensitive algorithm, RAND. The algorithm uses multiple iterations to eliminate the points and output the skyline. Each iteration consists of three scans. The first scan takes a sample from the points. The second scan replaces some of the points to increase the pruning power. The third scan eliminates the points and outputs the sample. This algorithm terminates in $O(dnm)$ comparisons, where m is the skyline cardinality. Although this method does not directly use a probability distribution model, the skyline cardinality is strongly related to the probability distribution.

The algorithms in this category rely on heuristic elimination and therefore the performance is sensitive to the skyline cardinality. Under dimensional independence assumption, the skyline cardinality is usually very small as discussed in [5, 8, 11] and these algorithms work well. The anti-correlated distributions considered in this paper usually result in extremely large skyline cardinality and these existing algorithms cannot terminate in their expected-time and likely fall back to the worst-case complexity $O(dn^2)$. The existing anti-correlated skyline algorithm [14] supports at most three dimensions, and therefore cannot satisify the requirement of applications.

2.3 Skyline cardinality

Under the sparseness and independence assumptions, the authors of [11] first propose that the skyline cardinality $\hat{S}_{d,n}$ is equal to the two-parameter harmonic number $H_{d-1,n}$. Alternatively, the authors of [8] evaluate skyline cardinality with a probabilistic model under the same assumptions as [11]. The probabilistic model in [8] is closely related to our proposed cardinality model. However, the model in [8] requires that the data should be dimensionally independent. Clearly, this requirement cannot be satisfied on anti-correlated data. In order to conquer this challenge, we introduce a new probabilistic skyline cardinality model in this paper. A recent work [23] proposes a kernel-based approach to approximate the skyline cardinality. This is a robust approach with nonparametric methods. However, it cannot fit our probabilistic model.

3. PRELIMINIARIES

Given a data space D defined by a set of d dimensions and a dataset P on D with cardinality n, a point $\bar{x} \in P$ can be represented as $\bar{x} = \{x_1, x_2, ...x_d\}$ where x_i is $\bar{x}$'s value on dimension i.

DEFINITION 1. *(Skyline and skyline cardinality) A point $\bar{x}$ dominates another point $\bar{y}$, denoted as $\bar{x} \prec \bar{y}$, if (1) on every dimension $i \in D$, $x_i \leq y_i$; and (2) on at least one dimension $j \in D$ $x_j < y_j$. The skyline $SKY_P \subseteq P$ is defined as a set of points in P which are not dominated by any other point in P. The points in a skyline are called skyline points and the skyline cardinality (i.e. output cardinality) is defined as the number of the skyline points.*

Let us consider a skyline operation over a set of n points with d dimensions. Let $S_{d,n}$ be the random variable which measures the skyline cardinality, and $\hat{S}_{d,n}$ denote the expected value of $S_{d,n}$.

Godfrey et al. [11, 12] propose a uniform independence model (UI) under which it is possible to analytically establish the skyline cardinality. This model applies the following assumptions on the input set.

DEFINITION 2. *UI model of the input dataset:*

- *Independence: the dimensions are statistically independent.*
- *Sparseness: for each dimension, there are not many duplicate values.*

- *Uniformity: the values on each dimension follow a uniform distribution.*

Additionaly, this model assumes under uniformity, without loss of generality, that any value is on the interval $(0, 1)$. The normalized method is also described in [11, 12].

We define an anti-correlated distribution by extending the UI model as follows.

DEFINITION 3. *Given a set of d-dimensional points under the UI model on the interval $(0, 1)$, an anti-correlated distribution with anti-correlation ratio c is obtained by removing the points $\bar{x}$ satisfying either $\sum_{i=1}^{d} x_i < d - 1$ or $\sum_{i=1}^{d} x_i > d - 1 + c$ where $0 \leq c \leq 1$.*

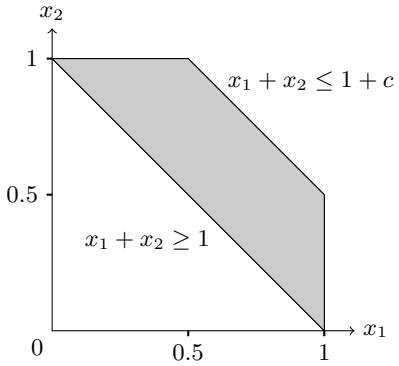


Figure 2: Anti-correlated distribution

An example in a 2D space is shown in Figure 2. All the points fall into the shaded area which is bound by the inequality $1 \leq x_1 + x_2 \leq 1 + c$. In general, the points under an anti-correlated distribution are close to the hyperplane $\sum_{i=1}^{d} x_i = d - 1$ and the anti-correlation ratio c defines the L1 norm distance to the hyperplane. The smaller the anti-correlation ratio c, the more serious the anti-correlation happens.

We further consider $S_{d,n,c}$ as a random variable which measures the skyline cardinality of the skyline operator on the anti-correlated distribution with anti-correlation ratio c. $\hat{S}_{d,n,c}$ denotes the expected value of $S_{d,n,c}$.

4. SKYLINE CARDINALITY ANALYSIS

We analyze the skyline cardinality in a step-by-step manner. We first explain the skyline cardinality with detailed examples in 2D and 3D spaces. Then we extend the obtained inequalities and estimations to a general d-dimensional space.

4.1 2D Case

THEOREM 1. *In a 2D space, the expected value $\hat{S}_{2,n,c}$ of the skyline cardinality satisfies:*

$$\hat{S}_{2,n,1} \leq \hat{S}_{2,n,c} \leq \hat{S}_{2,n,0} \quad (1)$$

$$\hat{S}_{2,n,0} = n \quad (2)$$

$$\hat{S}_{2,n,1} = \frac{n\sqrt{\pi}\Gamma(n)}{\Gamma(n+\frac{1}{2})} - 1 \approx \sqrt{n\pi} - 1 \quad (3)$$

where $\Gamma(n) = \int_0^\infty e^{-t} t^n dt$.

PROOF. The equations (1) and (2) are straightforward. We prove the equation (3) as follows. As shown in Figure 3,

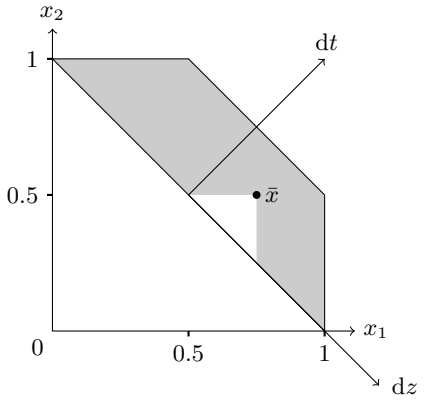


Figure 3: Skyline cardinality, 2D

a point is a skyline point if and only if all the other $n - 1$ points appear in the shaded area, and the probability $f(\bar{x})$ that a point appears in the shaded area is equal to the size of the area divided by the size of the whole area. Based on the similar idea of [8], the integral on the whole area provides the expected value of the skyline cardinality. That is:

$$\hat{S}_{2,n,1} = n \cdot 2 \iint_D f(\bar{x})^{n-1} \, \mathrm{d}\bar{x}$$

The above equation is solved by the integral on the (z, t) axis in Figure 3. That is:

$$\begin{aligned} \hat{S}_{2,n,1} =& n \cdot 2 \int_0^{\sqrt{2}/2} \int_0^{\sqrt{2}-2t} (\frac{1/2 - t^2}{1/2})^{n-1} dz dt \\ =& n \cdot 2 \int_0^{\sqrt{2}/2} \sqrt{2} (\frac{1/2 - t^2}{1/2})^{n-1} dt - 1 \end{aligned}$$

Replace t by $t = \frac{\sqrt{2x}}{2}$ then:

$$\hat{S}_{2,n,1} = n \cdot \int_0^1 (1-x)^{n-1} x^{-1/2} dx - 1$$

From Euler's Beta Integral:

$$\int_0^1 (1-t)^{b-1} t^{a-1} dt = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

we have:

$$\hat{S}_{2,n,1} = n \cdot \frac{\Gamma(\frac{1}{2})\Gamma(n)}{\Gamma(n+\frac{1}{2})} - 1 = \frac{n\sqrt{\pi}\Gamma(n)}{\Gamma(n+\frac{1}{2})} - 1$$

When n is sufficient large, together with the property of $\Gamma(n)$:

$$\lim_{n\to\infty} \frac{\sqrt{n}\Gamma(n)}{\Gamma(n+\frac{1}{2})} = 1$$

we have:

$$\hat{S}_{2,n,1} \approx \sqrt{n\pi} - 1$$

□

4.2 3D Case

THEOREM 2. *In a 3D space, the expected value $\hat{S}_{3,n,c}$ of the skyline cardinality satisfies:*

$$\hat{S}_{3,n,1} \leq \hat{S}_{3,n,c} \leq \hat{S}_{3,n,0} \quad (4)$$

$$\hat{S}_{3,n,0} = n \quad (5)$$

$$\hat{S}_{3,n,1} = \frac{n\Gamma(\frac{1}{3})\Gamma(n)}{\Gamma(n+\frac{1}{3})} - \frac{2n\Gamma(\frac{2}{3})\Gamma(n)}{\Gamma(n+\frac{2}{3})} + 1 \quad (6)$$

$$\approx \Gamma(\frac{1}{3})n^{\frac{2}{3}} - 2\Gamma(\frac{2}{3})n^{\frac{1}{3}} + 1 \quad (7)$$

where $\Gamma(n) = \int_0^\infty e^{-t}t^n dt$.

Additionally, the numerical approximation is:

$$\hat{S}_{3,n,1} \approx 2.679n^{\frac{2}{3}} - 2.708n^{\frac{1}{3}} + 1$$

PROOF. The equations (4) and (5) are straightforward. We provide the proof-sketch of the equations (6) and (7) as follows. Similar to Theorem 1, if $f(\bar{x})$ refers to the probability that a point does not dominate $\bar{x}$. The following integral provides the expected value of the skyline cardinality:

$$\hat{S}_{3,n,1} = n \cdot 6 \iiint_D f(\bar{x})^{n-1}\, \mathrm{d}\bar{x}$$

Let us imagine that there is a t-axis parallel to the vector $\{1,1,1\}$. A point is a skyline point if and only if none of the other $n-1$ points appear in the rectangular pyramid subspace which dominates this point. That is:

$$f(\bar{x}) = \frac{1/6 - \sqrt{3}/2 \cdot t^3}{1/6}$$

When the integral is on the t-axis, the size of the triangle on the plane which is vertical to the t-axis equals $\frac{3\sqrt{3}}{2}(\frac{\sqrt{3}}{3} - t)^2$. Then, we have:

$$\begin{aligned}
\hat{S}_{3,n,1} =& n \cdot 6 \int_0^{\sqrt{3}/3} (\frac{1/6 - \sqrt{3}/2 \cdot t^3}{1/6})^{n-1} \frac{3\sqrt{3}}{2}(\frac{\sqrt{3}}{3} - t)^2 dt \\
=& n \cdot 6 \int_0^{\sqrt{3}/3} (1 - 3\sqrt{3} \cdot t^3)^{n-1} \frac{\sqrt{3}}{2} dt \\
& - n \cdot 6 \int_0^{\sqrt{3}/3} (1 - 3\sqrt{3} \cdot t^3)^{n-1} 3t dt \\
& + n \cdot 6 \int_0^{\sqrt{3}/3} (1 - 3\sqrt{3} \cdot t^3)^{n-1} \frac{3\sqrt{3}}{2} t^2 dt \\
=& \frac{n\Gamma(\frac{1}{3})\Gamma(n)}{\Gamma(n+\frac{1}{3})} - \frac{2n\Gamma(\frac{2}{3})\Gamma(n)}{\Gamma(n+\frac{2}{3})} + 1
\end{aligned}$$

When n is sufficient large, together with the property of $\Gamma(n)$:

$$\lim_{n\to\infty} \frac{n^\alpha \Gamma(n)}{\Gamma(n+\alpha)} = 1$$

we have:

$$\hat{S}_{3,n,1} \approx \Gamma(\frac{1}{3})n^{\frac{2}{3}} - 2\Gamma(\frac{2}{3})n^{\frac{1}{3}} + 1$$

□

4.3 General Case

THEOREM 3. *The expected value $\hat{S}_{d,n,c}$ of the skyline cardinality in a d-dimensional space satisfies:*

$$\hat{S}_{d,n,1} \leq \hat{S}_{d,n,c} \leq \hat{S}_{d,n,0} \quad (8)$$

$$\hat{S}_{d,n,0} = n \quad (9)$$

$$\hat{S}_{d,n,1} = \sum_{k=1}^{d} (-1)^{k-1} \binom{d-1}{k-1} n \frac{\Gamma(\frac{k}{d})\Gamma(n)}{\Gamma(n+\frac{k}{d})} \quad (10)$$

$$\approx \sum_{k=1}^{d} (-1)^{k-1} \binom{d-1}{k-1} \Gamma(\frac{k}{d}) n^{1-\frac{k}{d}} \quad (11)$$

where $\Gamma(n) = \int_0^\infty e^{-t}t^n dt$ *for any constant natural number* $d \geq 2$.

PROOF. The proof-sketch of the equations (10) and (11) is nearly the same as that of the equations (6) and (7) in Theorem 2. The difference is replacing the rectangular pyramid subspace by a subspace in the d-dimensional space and replacing the triangle on the plane by a subspace on the $(d-1)$-dimensional hyperplane. Then, the proof of this theorem is immediate from the proof of Theorem 2. □

Theorem 3 describes the skyline cardinality on d-dimensional anti-correlated data. The lower bound of the skyline cardinality can be estimated by the polynomial in Equation (11). We observe that the degree of the polynomial is $n^{\frac{d-1}{d}}$. It means that the degree of the polynomial grows when the dimensionality increases. The lower bound of the degree of the polynomial is $n^{\frac{1}{2}}$ when $d = 2$.

THEOREM 4. *The expected value $\hat{S}_{d,n,c}$ of the skyline cardinality in d-dimensional space satisfies:*

$$O(n^{\frac{d-1}{d}}) \leq O(\hat{S}_{d,n,c}) \leq O(n)$$

for any constant natural number $d \geq 2$.

PROOF. For a constant natural number $d \geq 2$, this theorem is immediate from Theorem 3. □

In summary, the skyline operator on anti-correlated data is an operator with high output cardinality and this cardinality increases with the dimensionality. The lower and upper bounds of the skyline cardinality is proposed by equation (9) and (10) in Theorem 3 respectively. The lower bound can be estimated by the polynomial in equation (11).

Cost Analysis of Elimination Framework.

Theorem 3 establishes the lower and upper bounds of the skyline cardinality on anti-correlated distributions. It is well-known that the skyline cardinality in a d-dimensional space is $O(\log^{d-1} n)$ [5, 11], when the dimensions are independent. The high skyline cardinality on anti-correlated data results in significant performance downgrade of the existing algorithms.

The running time of the algorithms in the elimination category is sensitive to the skyline cardinality. Sarma et al. [20] claims that the algorithms [7, 12, 20] falls in $O(dmn)$ complexity where m is the skyline cardinality. Together with Theorem 3, the elimination algorithms answer skyline query in $O(dn^{\frac{3}{2}})$ and $O(dn^2)$ when the data is anti-correlated with $c = 1$ and $c = 0$ even in a 2D space, respectively. These two cost estimations clearly show that the performance of these algorithms decrease with the increasing of anti-correlation. As a result, these algorithms become impractical on anti-correlated data, even in low dimensional spaces.

5. ALGORITHMS

Before introducing our algorithms, we briefly summarize the elimination strategy. The algorithm first sorts the points $\bar{x}$ by a monotonic **scoring function** $F(\bar{x})$. Then the algorithm reads the points in sequence. For the k-th point, the algorithm conducts dominance check between this point and the temporary skyline points among the first $k-1$ points. If at least one of the temporary skyline points dominates the k-th point, the k-th point is eliminated. Otherwise, this point is inserted into the temporary skyline.

This framework achieves remarkable efficiency on dimensionally independent data. Since the skyline points with low scores on $F(\bar{x})$ can prune a majority of other points, most of the other points are pruned in a few dominance checks. As discussed in the previous section, anti-correlations result in a high skyline cardinality. This result causes two reasons which drastically downgrade the performance of the existing algorithms: (1) a huge number of temporary skyline points and (2) a low probability that a point can be eliminated. As a result, most of the points cannot be eliminated and have to be compared with a huge number of temporary skyline points.

5.1 Baseline Algorithms

We first propose a baseline algorithm to introduce our framework. The baseline algorithm is described in Algorithm 1. The function $F(\bar{x})$ must be a monotonic function. That is $F(\bar{x}) < F(\bar{y}) \Rightarrow \bar{y} \not\prec \bar{x}$.

Algorithm 1: **SkylineAC-Baseline**(P)

```
   Input  : P is the dataset;
   Output: S is the skyline set;
1  Sort P ascendingly by x_1;
2  S = ∅;
3  for read x̄ from P in sequence do
4  |  if exists ȳ ∈ S ∧ F(ȳ) ≤ F(x̄) then
5  |  |  for each ȳ ∈ S satisfying F(ȳ) ≤ F(x̄) do
6  |  |  |  if ȳ dominates x̄ then
7  |  |  |  |  goto Line 3;
8  |  |  |  end if
9  |  |  end for
10 |  |  S = S ∪ {x̄};
11 |  else
12 |  |  S = S ∪ {x̄};
13 |  end if
14 end for
```

This algorithm first sorts the points by the values in the first dimension. Then, the algorithm answers a skyline query in two steps: **determination** (line 4) and **elimination** (line 5−9). In the determination step, the algorithm quickly determines if a point is in the skyline and directly inserts the positive point into the skyline set S. Otherwise, the point is still possible to be either a skyline point or a non-skyline point. The algorithm tries to eliminate the point in the elimination step.

The implementation of the determination step is straightforward. Let us consider the L1 distance function $F(\bar{x}) = \sum_{i=2}^{d} x_i$ as an example. We maintain the minimal value $\min \sum_{i=2}^{d} y_i$ for all the points $\bar{y}$ in S. By comparing $\min \sum_{i=2}^{d} y_i$ with $\sum_{i=2}^{d} x_i$, the algorithm quickly determines whether the point can be directly inserted into the skyline set.

In the elimination step, we maintain S as a priority queue on $\sum_{i=2}^{d} y_i$. For each point $\bar{x}$, we scan S for any $\bar{y}$ satisfying $\sum_{i=2}^{d} y_i \leq \sum_{i=2}^{d} x_i$ and conduct dominance check between these points and $\bar{x}$. Since the points are sorted by the values in the first dimension beforehand, the points which are compared with $\bar{x}$ are bounded by two inequalities $y_1 \leq x_1$ and $\sum_{i=2}^{d} y_i \leq \sum_{i=2}^{d} x_i$. Compared with the traditional elimination framework with L1 distance $\sum_{i=1}^{d} x_i$ as the scoring function, all the points satisfying the above two inequality must satisfy the inequality $\sum_{i=1}^{d} y_i \leq \sum_{i=1}^{d} x_i$. Clearly, our proposed framework has a smaller search space than the traditional one. As a trade-off, we pay additional $O(\log n)$ cost per each skyline point to insert the skyline points into a priority queue. When the data is highly anti-correlated to a hyperplane $\sum_{i=1}^{d} x_i = K$ where K is a constant value, there is no doubt that $y_1 < x_1 \Leftrightarrow \sum_{i=2}^{d} y_i > \sum_{i=2}^{d} x_i$ and all the points can be determined. It guarantees the complexity $O(n \log n), O(dn), O(n \log n)$ for sort, determination and elimination respectively, when the data is highly anti-correlated to a $(d-1)$-dimensional hyperplane. Therefore, this algorithm is named as **AC-G-FD** (Anti-correlated skyline with guarantee and sorting the points by the first dimension) .

Name	$F(\bar{x})$
AC-G-FD	$\sum_{i=2}^{d} x_i$
AC-E-FD	$\sum_{i=2}^{d} \ln(x_i + 1)$
AC-P-FD	$-\sum_{i=2}^{d} \ln(1 - x_i)$

Table 1: the scoring functions for the baseline algorithms

As a case study, we integrate the traditional scoring functions into our framework. The most famous one is the entropy function which is firstly adopted in SFS by Chomicki et al. [9]. The intuitive idea is sorting by the first dimension and applying entropy function $F(\bar{x}) = \sum_{i=2}^{d} \ln(x_i + 1)$ on the other dimensions. This approach is shown as **AC-E-FD** (Anti-correlated skyline with entropy function and sorting the points by the first dimension) in Table 1.

Another alternative scoring function is the probability function. When $c = 1$, a point $\bar{x}$ dominates the hypercube with the volume $\prod_{i=1}^{d}(1 - x_i)$. Since this volume represents the probability that another point appears in the area which are dominated by the current point, we consider the volume as the pruning power of each point. Thus, the method **AC-P-FD** (Anti-correlated skyline with probablity function and sorting the points by the first dimension) in Table 1 sorts the points by the first dimension and organizes the priority queue by the probability function $F(\bar{x}) = -\sum_{i=2}^{d} \ln(1 - x_i)$ on the other dimensions. This probability function represents the descending order of volumes.

5.2 Scoring Functions on Two Clusters

We further optimize the determination and elimination framework by applying the scoring function on two clusters of dimensions. If we replace x_1 and $F(\bar{x})$ by the scoring functions $F_1(\bar{x})$ and $F_2(\bar{x})$ respectively, Algorithm 1 is directly generalized as the determination and elimination framework on two clusters. As shown in Algorithm 2, that is replacing x_1 by $F_1(\bar{x})$ in Algorithm 1 line 1 and replacing $F(\bar{y}) \leq F(\bar{x})$ by $F_2(\bar{y}) \leq F_2(\bar{x})$ in Algorithm 1 line 4 and 5. The two functions $F_1(\bar{x})$ and $F_2(\bar{x})$ must be monotonic.

That is $F_1(\bar{x}) < F_1(\bar{y}) \Rightarrow \bar{y} \not\prec \bar{x}$ and $F_2(\bar{x}) < F_2(\bar{y}) \Rightarrow \bar{y} \not\prec \bar{x}$. The implementation of this framework is the same as that of the baseline algorithms.

Algorithm 2: **SkylineAC-TwoClusters**(P)

```
1  Sort P ascendingly by F_1(x̄);
4  if exists ȳ ∈ S ∧ F_2(ȳ) ≤ F_2(x̄) then
5  |  for each ȳ ∈ S satisfying F_2(ȳ) ≤ F_2(x̄) do
9  |  end for
13 end if
```

Compared with the baseline algorithms, we observe that it is more efficient if we sort the points by the L1 distance on half dimensions $F_1(\bar{x}) = \sum_{i=1}^{\lfloor \frac{d}{2} \rfloor} x_i$ and eliminate the points by the L1 distance of the other half $F_2(\bar{x}) = \sum_{i=\lfloor \frac{d}{2} \rfloor+1}^{d} x_i$. This algorithm is named as **AC-G-HD** (Anti-correlated skyline with guarantee and sorting the points half dimensions) in Table 2. When the data is highly anti-correlated to a $(d-1)$-dimensional hyperplane, this algorithm ensures $F_1(\bar{x}) < F_1(\bar{y}) \Leftrightarrow F_2(\bar{x}) > F_2(\bar{y})$ for $c = 0$, and all the points can be determined. This property guarantees the complexity $O(n \log n), O(dn), O(n \log n)$ for sort, determination and elimination respectively, when the data is highly anti-correlated to a hyperplane $\sum_{i=1}^{d} x_i = K$ where K is a constant value.

Name	$F_1(\bar{x})$	$F_2(\bar{x})$
AC-G-HD	$\sum_{i=1}^{\lfloor \frac{d}{2} \rfloor} x_i$	$\sum_{i=\lfloor \frac{d}{2} \rfloor+1}^{d} x_i$
AC-E-HD	$\sum_{i=1}^{\lfloor \frac{d}{2} \rfloor} \ln(x_i+1)$	$\sum_{i=\lfloor \frac{d}{2} \rfloor+1}^{d} \ln(x_i+1)$
AC-P-HD	$-\sum_{i=1}^{\lfloor \frac{d}{2} \rfloor} \ln(1-x_i)$	$-\sum_{i=\lfloor \frac{d}{2} \rfloor+1}^{d} \ln(1-x_i)$

Table 2: the scoring functions on two clusters

Similar to the baseline algorithms, we extend this framework to integrate the traditional scoring functions. The algorithm **AC-E-HD** (Anti-correlated skyline with entropy function and sorting the points by half dimensions) adopts the entropy function in SFS by Chomicki et al. [9]. As shown in Table 2, the algorithm **AC-E-HD** sorts the points by $F_1(\bar{x}) = \sum_{i=1}^{\lfloor \frac{d}{2} \rfloor} \ln(x_i+1)$ and eliminates the points by $F_2(\bar{x}) = \sum_{i=\lfloor \frac{d}{2} \rfloor+1}^{d} \ln(x_i+1)$.

Another alternative algorithm **AC-P-HD** (Anti-correlated skyline with probability function and sorting the points by half dimensions) sorts the points by the probability function $F_1(\bar{x}) = -\sum_{i=1}^{\lfloor \frac{d}{2} \rfloor} \ln(1-x_i)$ on half dimensions and eliminates the points by the same function $F_2(\bar{x}) = -\sum_{i=\lfloor \frac{d}{2} \rfloor+1}^{d} \ln(1-x_i)$ on the other half dimensions.

5.3 Indexing Techniques

In this subsection, we propose indexing techniques to improve the pruning power of the determination and elimination framework. The key idea is to increase the number of clusters and reduce the number of dimensions which are covered by each scoring function. The indexed framework is shown in Algorithm 3. This framework further replaces $F_2(\bar{y}) \leq F_2(\bar{x})$ by **satisfying Condition**$(\bar{y}, \bar{x})$ in Algorithm 2 line 4 and 5. The **Condition**$(\bar{y}, \bar{x})$ is defined as $F_2(\bar{y}) \leq F_2(\bar{x}) \wedge F_3(\bar{y}) \leq F_3(\bar{x}) \wedge ... \wedge F_k(\bar{y}) \leq F_k(\bar{x})$ where k is the number of clusters. The framework in the previous subsection can be considered as a special case when $k = 2$.

We first consider $k = 3$ as a representative case. The scoring functions on three clusters of dimensions are described in Table 3. In such case, the **Condition**$(\bar{y}, \bar{x})$ is equivalent to $F_2(\bar{y}) \leq F_2(\bar{x}) \wedge F_3(\bar{y}) \leq F_3(\bar{x})$. Similar to the two-clusters framework, we implement two steps in the query processing: determination (line 4) and elimination (line $5-9$) in Algorithm 3. In the determination step, the problem is similar to the 3D skyline problem. If we consider $F_1(\bar{x})$, $F_2(\bar{x})$ and $F_3(\bar{x})$ as three virtual dimensions, we modify the algorithm by Kung et al. [16] to implement this step. Since the data has already been sorted by $F_1(\bar{x})$, we read the data in sequence and maintain the pairs $\{F_2(\bar{x}), F_3(\bar{x})\}$ in an AVL tree with $F_2(\bar{x})$ as the key. For each tuple $\{F_2(\bar{x}), F_3(\bar{x})\}$, we retrieve the highest lower bound $F_2(\bar{y})$ of $F_2(\bar{x})$ from the AVL tree. If such lower bound $F_2(\bar{y})$ does not exist or $F_3(\bar{x}) < F_3(\bar{y})$, $\bar{x}$ is determined as a skyline point; the current tuple $\{F_2(\bar{x}), F_3(\bar{x})\}$ is inserted into the AVL tree and the other tuples $\{F_2(\bar{z}), F_3(\bar{z})\}$ satisfying $F_2(\bar{x}) \leq F_2(\bar{z}) \wedge F_3(\bar{x}) \leq F_3(\bar{z})$ are removed. This algorithm terminates in $O(n \log n)$ complexity. The elimination step is equivalent to the range query in a 2D space. We straightforwardly deploy QuadTree [10] to retrieve the tuples $\bar{y}$ which satisfy **Condition**$(\bar{y}, \bar{x})$.

When the number of clusters increases to four and more, the algorithm requires high dimensional spatial indexes. We use Octree [13] for $k = 4$ and kd-tree [2] for $k > 4$. Clearly, both the pruning power and the indexing cost overhead increase with the number of clusters. The indexing techniques with more clusters are suitable for the data with more anti-correlated dimensions.

Algorithm 3: **SkylineAC-Index**(P)

```
1  Sort P ascendingly by F_1(x̄);
4  if exists ȳ ∈ S∧ ȳ satisfying Condition(ȳ, x̄) then
5  |  for each ȳ ∈ S satisfying Condition(ȳ, x̄) do
9  |  end for
13 end if
```

5.4 Automatically Clustering Dimensions

The anti-correlation is well-known in many applications. For instance, the real estate price and the land tax are anti-correlated to the land size (prefer larger to smaller), the building size (prefer larger to smaller) and the distance to the city. Expert users can manually partition the anti-correlated dimensions into different clusters. However, non-expert users may not be able to discover the underlying anti-correlation of data. We propose an efficient algorithm to help the users to detect anti-correlation and effectively partition the anti-correlated dimensions into several clusters.

In the first step, we conduct a quick anti-correlation detection in $O(dn)$ time. For each dimension $i = 1, 2, ..., d$, we calculate the maximum value U_i and the minimum value L_i. Similar to the idea proposed by Bentley et al. [4], we assume that there is a virtual point $\bar{x} = \{x_1, x_2, ..., x_d\}$ where $x_i = (\ln n/n)^{1/d}(U_i - L_i) + L_i$. If there exists a point $\bar{y}$ in the dataset P which dominates the virtual point $\bar{x}$, we adopt the traditional algorithms in the elimination category. In this

Name	$F_1(\bar{x})$	$F_2(\bar{x})$	$F_3(\bar{x})$
AC-G-3D	$\sum_{i=1}^{\lfloor \frac{d}{3} \rfloor} x_i$	$\sum_{i=\lfloor \frac{d}{3} \rfloor+1}^{\lfloor \frac{2d}{3} \rfloor} x_i$	$\sum_{i=\lfloor \frac{2d}{3} \rfloor+1}^{d} x_i$
AC-E-3D	$\sum_{i=1}^{\lfloor \frac{d}{3} \rfloor} \ln(x_i+1)$	$\sum_{i=\lfloor \frac{d}{3} \rfloor+1}^{\lfloor \frac{2d}{3} \rfloor} \ln(x_i+1)$	$\sum_{i=\lfloor \frac{2d}{3} \rfloor+1}^{d} \ln(x_i+1)$
AC-P-3D	$-\sum_{i=1}^{\lfloor \frac{d}{3} \rfloor} \ln(1-x_i)$	$-\sum_{i=\lfloor \frac{d}{3} \rfloor+1}^{\lfloor \frac{2d}{3} \rfloor} \ln(1-x_i)$	$-\sum_{i=\lfloor \frac{2d}{3} \rfloor+1}^{d} \ln(1-x_i)$

Table 3: the scoring functions on three clusters

case, the algorithm terminates in total $O(dn)$ time [4], if the data is not anti-correlated.

If none of the points dominates the virtual point, we convert the values into ranks in each dimension . This step is implemented by sorting the points by each dimension and replacing the value by its rank.

We define $rank(x_i)$ as the rank of $\bar{x}$ on the i-th dimension. The dimension clustering algorithm is based on the minimal sum $minrank(D_1, D_2, ..., D_m)$ of the ranks on a set of dimensions.

DEFINITION 4. *Given a set $\{D_1, D_2, ..., D_m\}$ of m dimensions, the $minrank(D_1, D_2, ..., D_m)$ function is defined as:*

$$minrank(D_1, D_2, ..., D_m) = \min\{\sum_{i=1}^{m} rank(x_{D_i}) | \bar{x} \in P\}$$

Computing the rank of each dimension takes $O(dn \log n)$ time. Since the data has already been known to be anti-correlated in this step, the skyline cardinality $\hat{S}_{d,n,c}$ falls into $O(n^{\frac{d-1}{d}}) \leq O(\hat{S}_{d,n,c}) \leq O(n)$ by Theorem 4. Hence, the ranking cost is much less than the elimination cost which is inbetween $O(dn^{1+\frac{d-1}{d}})$ and $O(dn^2)$.

We describe the dimension clustering algorithm by explanation. We first predefine a set $\{\theta_2, \theta_3, ..., \theta_d\}$ of thresholds for the $minrank$ function to determine the number of indexed dimensions, where $\theta_m = \frac{n(m-1)}{2}$ for $m = 2, 3, ..., d$ from Definition 3.

Initially, we calculate the $minrank(1, 2, ..., d)$ on all the dimensions. If $minrank(1, 2, ..., d) > \theta_d$, we consider that all the dimensions are anti-correlated, and thus, we begin to build the index with d dimensions. Otherwise, we remove a dimension from the original d dimensions and calculate the $minrank(D_1, D_2, ..., D_{d-1})$ on the rest $d-1$ dimensions. Clearly, there exist $d-1$ possible removals and we choose the one with the highest $minrank(D_1, D_2, ..., D_{d-1})$. If the $minrank(D_1, D_2, ..., D_{d-1}) > \theta_{d-1}$, we start to build the index on these $d-1$ dimensions. Otherwise, the algorithm recursively removes another dimension from $\{D_1, D_2, ..., D_{d-1}\}$ and chooses the highest $minrank(D_1, D_2, ..., D_{d-2})$ until either $minrank(D_1, D_2, ..., D_m) > \theta_m$ or $m = 1$.

There are two possible cases when the above algorithm terminates: (1) the algorithm finds a set of dimensions satisfying $minrank(D_1, D_2, ..., D_m) > \theta_m$ or (2) $m = 1$. When $m = 1$, we consider the data is not anti-correlated and deploy existing algorithms (e.g. SFS [9], OSP [22] and etc) to solve the problem. In the follows, we discuss the case when $minrank(D_1, D_2, ..., D_m) > \theta_m$ for $m \geq 2$.

When we have m anti-correlated dimensions and we have to partition the m dimensions into k clusters with nearly the same size, the i-th cluster owns s_i dimensions which satisfies $\sum_{i=1}^{k} s_i = m$ and $\max(s_i) - \min(s_i) \leq 1$. For example, when $m = 11$ and $k = 3$, there exists a possible partition that $s_1 = 3$, $s_2 = 4$ and $s_3 = 4$. Among the m dimensions $\{D_1, D_2, ..., D_m\}$, we remove a dimension and calculate the $minrank(D_1, D_2, ..., D_{m-1})$ on the rest $m-1$ dimension. There are m possible removals $D_1, D_2, ..., D_m$ and we choose the one with the highest $minrank(D_1, D_2, ..., D_{m-1})$. Assuming that D_i is the removed dimension with the highest $minrank(D_1, D_2, ..., D_{m-1})$, we put D_i into the cluster j which has the highest capacity s_j. If there exists more than one cluster which have the same s_j, we randomly choose one of them. Then, we set $s_j = s_j - 1$. Recursively, we remove another dimension from $\{D_1, D_2, ..., D_{m-1}\}$ and repeat this procedure until all the dimensions are clustered. In summary, this algorithm removes the dimensions one-by-one, heuristically chooses the most anti-correlated $m-1$ dimensions from each m dimensions and distributes them into different clusters. If the data is anti-correlated, this algorithm takes $O(dn \log n)$ time to convert the values into ranks and $O(d^3 n)$ time to partition the dimensions. Otherwise, it terminates in the anti-correlation detection step which takes $O(dn)$ time.

6. EXPERIMENTS

We study the performance of our framework in this section. The experiments are performed on an Intel Xeon 2.4GHz processor running Linux operating system.

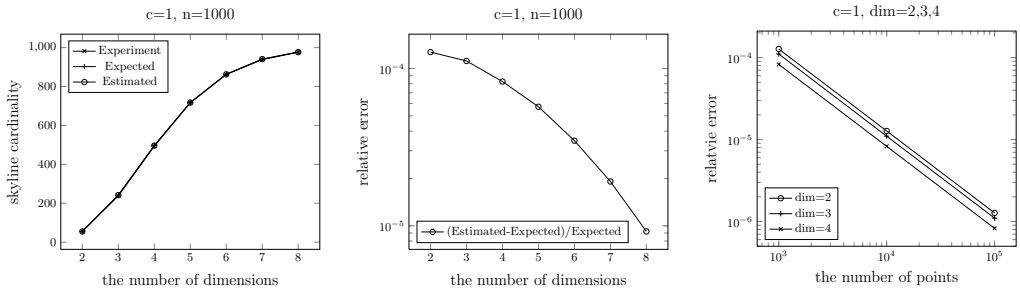
The benchmark synthetic anti-correlated datasets vary the size from $1k$ points to $100k$ data points with respect to $2-8$ dimensions. The dataset in default settings contains $10k$ points, All the datasets obey the criteria in Definition 3. The anti-correlation ratio c varies from 0.01 to 1. We consider the datasets with $c = 0.01$, $c = 0.1$ and $c = 1$ as strong anti-correlated datasets, medium anti-correlated datasets and weak anti-correlated datasets, respectively.

The NBA dataset contains $17k$ 13-dimensional data points, each of which corresponds to the statistics of an NBA player's performance in 13 aspects (such as points scored, rebounds, assists, field goals made, etc). However, we only consider 5-7 dimensions among 13 statistics in our evaluation since others may be missing in some records.

6.1 Skyline Cardinality

Figure 4 (a) illustrates the skyline cardinalities which are obtained from **Experiment** (average value of 16 runs), **Expected** value (equation (10) in Theorem 3) and polynomial **Estimated** value (equation (11) in Theorem 3)). The result evidences the correctness of Theorem 3.

Figure 4 (b) reports the accuracy of the polynomial estimation equation (10) in Theorem 3 against the number of dimensions. We report the accuracy by the relative error $\frac{\text{Estimated}-\text{Expected}}{\text{Estimated}}$. When $n = 1000$, the relative errors of the estimation are 1.3×10^{-4} and 9.2×10^{-6} for $d = 2$ and $d = 8$, respectively. We observe an increase of the accuracy when increasing the number of dimensions.



(a) Verifying the correctness (b) Accuracy of the estimation against the number of dimensions (c) Accuracy of the estimation against the number of points

Figure 4: Evaluation of the estimation in Theorem 3

Figure 4 (c) describes the accuracy against the number of points. The accuracy is represented by relative errors $\frac{\text{Estimated}-\text{Expected}}{\text{Estimated}}$. The result clearly indicates that the relative error synchronously decreases with the increasing of the number of points. The relative errors are about 10^{-4}, 10^{-5} and 10^{-6} for $n = 1000$, $n = 10000$ and $n = 100000$, respectively.

Figure 5 presents the experimental skyline cardinalities for different anti-correlation ratio c. The smaller the c value is, the more serious the anti-correlation will be. We reports the cardinality as per two different sets of c value, varying c linearly for $c = 0.2$, 0.4, 0.6, 0.8 and 1.0 and varying c logarithmically for $c = 0.01$, 0.05, 0.1, 0.5 and 1.0. Figures 5 (a) and (c) indicate that the skyline cardinality is close to the estimated lower bound when $c \geq 0.4$. In contrast, Figures 5 (b) and (d) illustrate that the skyline cardinality is approaching to the upper bound n when $c \leq 0.1$ and $d \geq 5$. Therefore, we evaluate the efficiency of the algorithms by logarithmically varying c values to explore the effect of anti-correlations.

6.2 Analysis on the Scoring Functions

We first evaluate the scoring functions and the dimensional clustering methods on anti-correlated data. For this purpose, we consider six algorithms which combine three scoring functions and two dimensional clustering methods. The three scoring functions are described as follows.

- AC-G-*: L_1 distance function.
- AC-E-*: Entropy function.
- AC-P-*: Probability function.

The two dimensional clustering methods includes:

- AC-*-FD: Baseline method with two clusters and sorting the points by the first dimensions.
- AC-*-HD: Our proposed dimensional clustering method with two clusters and sorting the points by the scoring function on half dimensions.

The Effect of Anti-Correlation Ratio c. We report the effect of the anti-correlation ratio c by logarithmically varying c from 0.01 to 1. The performance is evaluated by the total response time on $100k$ points. In Figure 6, the result illustrates the effect of anti-correlation ratio c on different dimensionalities.

In 3D and 4D spaces as shown in Figures 6 (a) and (b), the L_1 distance algorithms, AC-G-FD and AC-G-HD, outperform the entropy function algorithms, AC-E-FD and AC-E-HD while these two categories significantly outperform the probability function algorithms, AC-P-FD and AC-P-HD. If we rank the algorithm categories by their performance, this rank will be AC-G, AC-E and AC-P from the best to the worst.

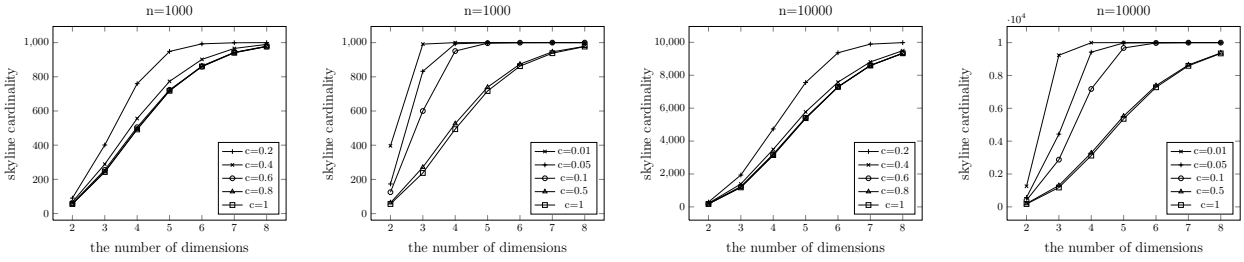
We report the performance evaluation of the algorithms in 5D and 6D spaces in Figures 6 (c) and (d), respectively. AC-G-HD is a clear winner under all the parameters. However, the second place depends on the anti-correlation ratio c. When $c = 0.01$ where the data is highly anti-correlated, AC-G-FD is in the second place since it is an algorithm with performance guarantee on highly anti-correlated data. AC-E-HD achieves the best performance among the others except AC-G-HD for $c \geq 0.1$. This result indicates that the functions with clustering on half dimensions (algorithms with suffix HD) are more effective than simply sorting the points by the first dimension.

The Effect of Dimensionality. We describe the effect of the number of dimensions in Figure 7. There is no doubt that AC-G-HD outperforms the others under all the settings. Hence, we focus on analyzing how the dimensionality affects the performance.

All the algorithms show a performance drop with the increase of dimensionality. However, as shown in Figures 7 (a), this performance drop slows down at $d = 3$ on the strong anti-correlated dataset where $c = 0.01$. It indicates that the performance is stable to the dimensionality on highly anti-correlated data. This result coincides with the cardinality upper bound $O(n)$ which is constant to the number of dimensions when $c = 0$. In contrast, the total response time continuously increases on the weak anti-correlated dataset where $c = 1$ in Figures 7 (c). Since the performance of skyline algorithms is affected by the skyline cardinality, this result coincides with the cardinality lower bound $O(n^{\frac{d-1}{d}})$ when $c = 1$. The experimental result in Figures 7 (b) illustrates a compromising situation on the medium anti-correlated dataset. The total response time increases with the dimensionality up to five dimensions, and thereafter the performance is stable.

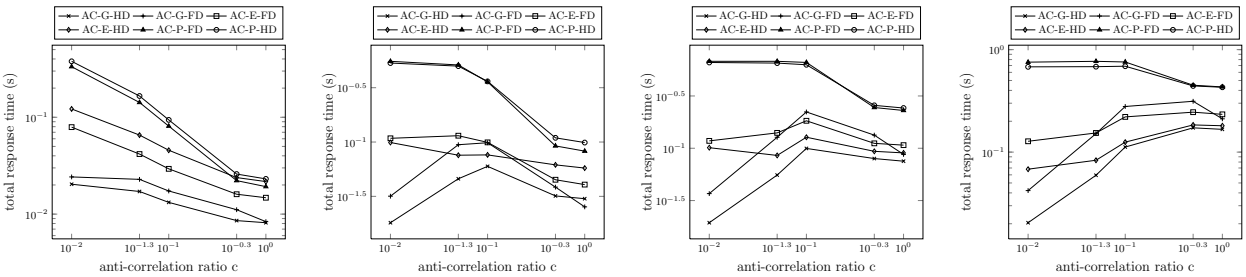
6.3 Algorithm Efficiency

In the algorithm efficiency evaluation, we focus on generic skyline algorithms in which preprocessing is not required. Since the L_1 distance function outperforms all the other scoring functions in the previous experimental study, we



(a) linearly (b) logarithmically (c) linearly (d) logarithmically

Figure 5: Experimental skyline cardinalities for different anti-correlation ratio c



(a) dim=3 (b) dim=4 (c) dim=5 (d) dim=6

Figure 6: Effect of anti-correlation ratio c

consider the L_1 distance function as our choice among all the scoring functions. The L_1 distance function is integrated with our proposed indexing techniques and automatically clustering algorithm. The proposed algorithm is named SOAD (Skyline Operator on Anti-correlated Distributions). The proposed algorithms and the state-of-the-art algorithms for case study are described as follows:

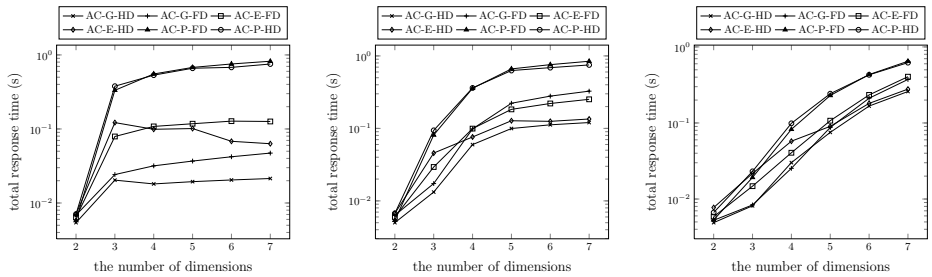
- SOAD: Our proposed algorithm in this paper.
- SFS/LESS: Skyline with presorting (SFS) algorithm is proposed by Chomicki et al. [9]. We consider SFS as a baseline algorithm. There is an optimized verison of SFS, namely LESS (linear elimination sort for skyline) [12] which improves the sorting efficiency of SFS. We will soon demonstrate that the filtering cost is the dominating cost of the skyline algorithms on anti-correlated data. In such case, SFS and LESS become the same algorithm.
- OSP: The object-based space partitioning approach (OSP) is proposed by Zhang et al. [22]. The index is based on the leftchild/ right-sibling tree. We implement the OSPSOnPartitioningFirst version of this algorithm. This algorithm can be faster than the alternatives such as OSPSOnSortingFirst in [22] and BSkyTree in [17], because it avoids the cost of selecting the point for partitioning and attempts to prune other points as soon as skyline points are found in their dominating partitions. This method also beats LESS [12] and SaLSa [1] under a wide range of settings.

Sorting Cost vs. Filtering Cost. The existing study [12] shows that the sorting cost is usually the major cost of the skyline algorithms when the dimensions are statistically independent. We explore the sorting cost and the filtering cost on anti-correlated data and report some interesting observations.

In Figures 8 (a) and (b), we evaluate the sorting cost and the filtering cost of SFS on the anti-correlated data by varying the anti-correlation ratio c among 0.01, 0.1 and 1. Clearly, the sorting cost is determined by the number of points. This cost is not affected by the number of dimensions or the anti-correlation ratio. In contrast, the filtering cost increases with the number of dimensions and decreases with the anti-correlation ratio c. It is worth noting that a small anti-correlation ratio c means that the data is highly anti-correlated. The result indicates that the filtering cost is much higher than the sorting cost even when the points are in a 2D space. In a 4D space, the filtering cost dominates at least 90% of the total cost for $n \geq 100k$ with arbitrary anti-correlation ratio c. Additionally, this result also suggests that the optimized version, LESS [12], which improves the sorting algorithm of SFS can be discarded in our experiment.

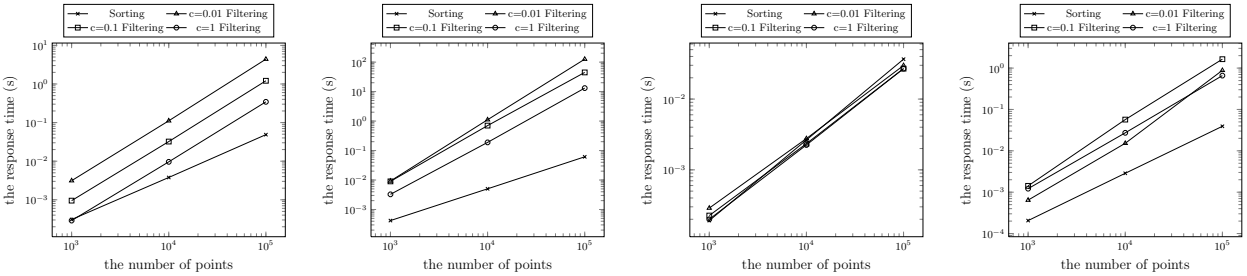
In Figures 8 (c) and (d), we compare the sorting cost to the filtering cost of our proposed algorithm. In a 2D space, the filtering cost is slightly lower than the sorting cost. As the filtering cost increases with the number of dimensions, the filtering cost becomes the major cost when the number of dimensions increases to four and more.

Comparison on Synthetic Datasets. As shown in Figure 9, the algorithms are evaluated by the total response time on the synthetic datasets. We consider the datasets with three to six dimensions in each figure respectively. The anti-correlation ratio c varies from 0.01 to 1. SOAD clearly outperforms the other two algorithms. The maximum performance gap always appears at $c = 0.01$ for all the numbers of dimensions. We observe a clear increasing performance gap between SOAD and the other two algorithms when the anti-correlation becomes strong (i.e. the anti-correlation ratio c decreases). The result demonstrates the effectiveness of the proposed techniques on anti-correlated data.



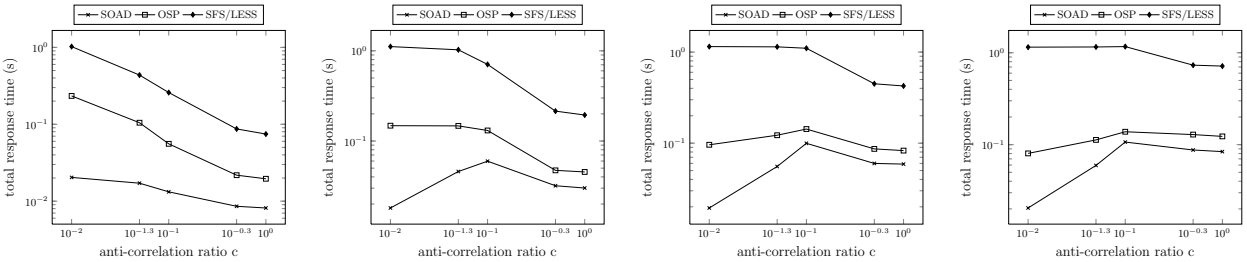
(a) Strong anti-correlated (b) Medium anti-correlated (c) Weak anti-correlated

Figure 7: Effect of the number of dimensions



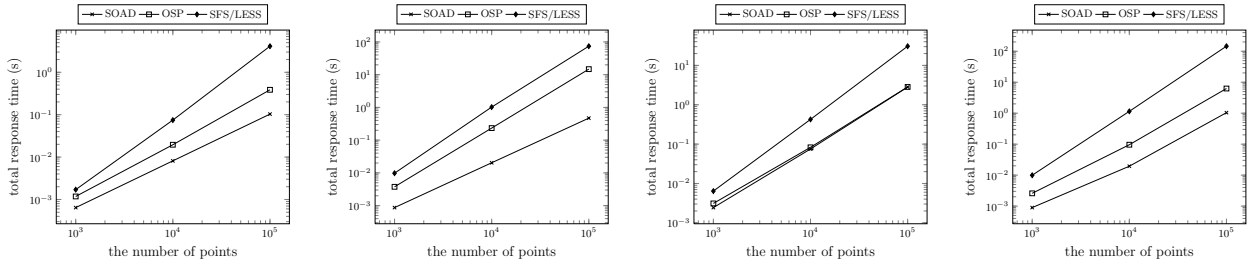
(a) SFS/LESS, dim=2 (b) SFS/LESS, dim=4 (c) SOAD, dim=2 (d) SOAD, dim=4

Figure 8: Sorting cost vs. filtering cost



(a) dim=3 (b) dim=4 (c) dim=5 (d) dim=6

Figure 9: Comparison with the state-of-the-art methods



(a) Weak anti-correlated, $dim = 3$ (b) Strong anti-correlated, $dim = 3$ (c) Weak anti-correlated, $dim = 5$ (d) Strong anti-correlated, $dim = 5$

Figure 10: Scalability regarding the number of points

Scalability. The scalability is studied in Figure 10. We choose the 3D and 5D datasets as the representative datasets. The anti-correlation ratio c varies among 0.01, 0.1 and 1. The results indicate that SOAD is the best choice. SFS/LESS is worst on the scalability, while OSP achieves comparable scalability on the weak anti-correlated datasets. The performance gap in between SOAD and the other two algorithms grows up with the dataset size on the strong anti-correlated datasets. When the number of points increases to $100k$, SOAD is more than an order and two orders of magnitude faster than OSP and SFS/LESS respectively.

Experiments on Real Datasets. We also conduct experiments on NBA datasets. As shown in Table 4, the skyline cardinalities are 817, 2061 and 3135 for $Dim = 5, 6$ and 7, respectively. Altough the anti-correlation on NBA datasets is weak, our proposed algorithm SOAD still significantly outperforms the existing algorithms SFS/LESS and OSP.

	Dim=5	Dim=6	Dim=7
SOAD	0.0063	0.0238	0.0450
OSP	0.0166	0.0328	0.0554
SFS/LESS	0.0146	0.0638	0.1257
Skyline cardinality	817	2061	3135

Table 4: RESPONSE TIME (SEC) ON NBA DATA

7. CONCLUSION

In this paper, we have studied the skyline operator on anti-correlated distributions. To tackle this problem, we establish a probabilistic model for anti-correlated distributions. Following the existing work, the d-dimensional points are close to a hyperplane in an anti-correlated distribution. We introduce the anti-correlation ratio c to define the maximum distance from the points to the hyperplane. Based on the proposed model, we prove the lower bound and upper bound of the expected skyline cardinality together with the effective polynomial estimation. Since the performance of the existing work is impractical on anti-correlated distributions, we also develop efficient algorithms to compute skyline on anti-correlated distributions. In contrast to the existing work which rely on heuristic elimination, our proposed techniques not only effectively eliminate non-promising points but also efficiently determine skyline points. As a result, our approach achieves significantly performance improvement over the existing work. The comprehensive experimental evaluation shows our algorithm outperforms the state-of-the-art algorithms on both the real NBA player datasets and the benchmark synthetic datasets under a wide range of settings. The performance gap is up to two orders of magnitude.

There are clearly many directions for future work. Our analysis is based on the anti-correlation to a hyperplane. It is meaningful to analyze and model other distributions of real world applications. It is also an interesting direction to develop efficient algorithms for these applications.

8. REFERENCES

[1] I. Bartolini, P. Ciaccia, and M. Patella. Efficient sort-based skyline evaluation. *ACM Trans. Database Syst.*, 33(4), 2008.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[3] J. L. Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980.

[4] J. L. Bentley, K. L. Clarkson, and D. B. Levine. Fast linear expected-time algorithms for computing maxima and convex hulls. *Algorithmica*, 9(2):168–183, 1993.

[5] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the average number of maxima in a set of vectors and applications. *J. ACM*, 25(4):536–543, 1978.

[6] J. L. Bentley and M. I. Shamos. Divide and conquer for linear expected time. *Inf. Process. Lett.*, 7(2):87–91, 1978.

[7] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[8] S. Chaudhuri, N. N. Dalvi, and R. Kaushik. Robust cardinality and cost estimation for skyline operator. In *ICDE*, 2006.

[9] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *ICDE*, pages 717–719, 2003.

[10] R. A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 1974.

[11] P. Godfrey. Skyline cardinality for relational processing. In *FoIKS*, pages 78–97, 2004.

[12] P. Godfrey, R. Shipley, and J. Gryz. Algorithms and analyses for maximal vector computation. *VLDB J.*, 16(1):5–28, 2007.

[13] C. L. Jackins and S. L. Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 14(3):249 – 270, 1980.

[14] H. Köhler and J. Yang. Computing large skylines over few dimensions: The curse of anti-correlation. In *APWeb*, pages 284–290, 2010.

[15] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, pages 275–286, 2002.

[16] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476, 1975.

[17] J. Lee and S. W. Hwang. Bskytree: scalable skyline computation using a balanced pivot selection. In *EDBT*, pages 195–206, 2010.

[18] X. Lin, Y. Yuan, W. Wang, and H. Lu. Stabbing the sky: Efficient skyline computation over sliding windows. In *ICDE*, pages 502–513, 2005.

[19] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD Conference*, pages 467–478, 2003.

[20] A. D. Sarma, A. Lall, D. Nanongkai, and J. Xu. Randomized multi-pass streaming skyline algorithms. *PVLDB*, 2(1):85–96, 2009.

[21] C. Sheng and Y. Tao. On finding skylines in external memory. In *PODS*, pages 107–116, 2011.

[22] S. Zhang, N. Mamoulis, and D. W. Cheung. Scalable skyline computation using object-based space partitioning. In *SIGMOD Conference*, pages 483–494, 2009.

[23] Z. Zhang, Y. Yang, R. Cai, D. Papadias, and A. K. H. Tung. Kernel-based skyline cardinality estimation. In *SIGMOD Conference*, pages 509–522, 2009.