Sampling-Based Approximate Skyline Calculation on Big Data

Xingxing Xiao\textsuperscript{1} and Jianzhong Li\textsuperscript{2}

\textsuperscript{1}xiaoxx@hit.edu.cn
\textsuperscript{2}lijzh@hit.edu.cn
Harbin Institute of Technology, Harbin, Heilongjiang 150001, China

Abstract. The existing algorithms for processing skyline queries cannot adapt to big data. The paper proposed two approximate skyline algorithms based on sampling. The first algorithm obtains a size $m$ sample and computes the approximate skyline on the sample. The running time of the algorithm is $O(m \log d - 2 n)$ in the worst case and $O(m)$ in the average case. The expected error of the algorithm is almost independent of the relation size, and the standard deviation is relatively small. The error of the first algorithm is small enough with a moderate $m$. The second algorithm returns an approximate skyline result with a given error bound $\epsilon$. The time complexity of the algorithm is $O(m^3 \log m)$, where $m$ is almost unaffected by the relation size. Experiments verify the error analysis of the first algorithm and show that the second algorithm is much faster than the existing skyline algorithms.

Keywords: Sampling · Skyline · Approximation · Big Data

1 Introduction

Skyline query is important in many applications involving multi-criteria decision making. Given a relation $T(A_1, A_2, ..., A_d)$ and a set of skyline criteria $C \subseteq \{A_1, ..., A_d\}$, a skyline query on $T$ is to find the largest subset of $T$ such that $t$ is not dominated by any tuple in $T$, where $t'$ dominates $t$, written as $t' \prec t$, means that $t'.A_i \leq t.A_i$ for all $A_i \in C$ and there is an attribute $A_j \in C$ such that $t'.A_j < t.A_j$. Hereafter, we use $t.A_i$ to express the value of tuple $t \in T$ on attribute $A_i$. Skyline queries can also be defined using $\geq$ and $\succ$. Without loss of generality, this paper only consider the skyline queries defined by $\leq$ and $\prec$. The answers to a skyline query are all the potentially best tuples to users, and skyline queried provide good mechanisms for merging user’s preferences into queries.

Studies on skyline originated in theoretical computer science area in the last century. Skyline was called as the set of maximals or the pareto set in that time. Many algorithms for finding the maximals were proposed \cite{3415}. The lowest time complexity of these algorithms is $O(n \log d - 2 n)$ in the worst case, and $O(n)$ in the average case. However, all the algorithms are based on Divide&Conquer strategy and assume that their input tuples are stored in the main memory.
Borzsony first introduced the skyline query to the database field [5]. It attracted considerable attention to design efficient algorithms for processing skyline queries on relations stored in external storage. Many algorithms have been proposed [2, 5, 7, 11]. The lowest time complexity of the algorithms is $O(n^2)$ in the worst case, and $O(n)$ in the average case.

Nowadays, big data is coming to the force in a lot of applications. Processing a skyline query on big data in more than linear time is by far too expensive and often even linear time may be too slow. Thus, the sublinear time algorithm for processing skyline queries becomes a highly concerned research subject. Many index-based algorithms for processing skyline queries have been proposed to achieve the sublinear running time in the average case [5, 12, 14, 16, 20, 22]. However, all the algorithms have serious limitations. Firstly, the algorithms require much time for pre-computation, which is at least $\Omega(n)$. Secondly, they need expensive extra space overhead for indexes. Thirdly, there is much overhead to maintain indexes while the input relations are updated.

Approximation computation of skylines is the only way to break through the three limitations. Fortunately, approximate skyline results are enough in many applications. An example of the skyline query is to find restaurants near the workplace that provide delicious foods and excellent services. To get the answer quickly, users can accept approximate skyline results that are the good restaurants but not the best ones. Actually, users prefer to get an approximate result in seconds rather than an exact result in hours or more in many applications.

There has been many researches on approximation algorithms for skyline queries [13, 17, 18, 21, 23], but their goal is to reduce the skyline size and approximate the best subset of $k$ input tuples to represent the skyline under various measures. Moreover, they have higher running time than the precise algorithms for processing skyline queries.

In this paper, we propose two sampling-based approximation algorithms for processing skyline queries on big data. The proposed algorithms don’t need any extra space or pre-computation overhead. Viewing the skyline as a covering, the error of a approximation algorithm is defined as $|\frac{DN(Sky) - DN(\tilde{Sky})}{DN(Sky)}|$, where $DN(Sky)$ is the number of tuples dominated by the approximate result $\tilde{Sky}$, and $DN(Sky)$ is the number of tuples dominated by the exact result $Sky$. If $\frac{DN(Sky) - DN(\tilde{Sky})}{DN(Sky)} \leq \epsilon$, then $\epsilon$ is called as the error bound of the approximation algorithm.

The first proposed algorithm draws a random sample from the input relation at the beginning, and then computes the approximate skyline on the sample. The algorithm has two advantages. First, the expected error of the algorithm is almost independent of the input relation size. Second, the standard deviation of the error is relatively small. These advantages have been verified in experiments.

The second algorithm returns an approximate skyline result efficiently with a given error bound. To guarantee the error bound, the total sample size required by the algorithm is almost a constant relative to the input relation size. The algorithm first draws an initial sample, and then computes the approximate
skyline on the sample. Afterwards, the algorithm verifies the error of the current result. If the error cannot be guaranteed to reach the error bound, the algorithm doubles the sample size and repeats the above process. Otherwise, the algorithm terminates.

Extensive experiments show that the second algorithm involves only constant number of tuples, and is much faster than the existing skyline algorithms. The full experimental results can be found in the online full version of this paper [24].

The main contributions of the paper are listed below.

(1). A baseline approximation algorithm for processing skyline queries is proposed, which is based on a sample of size \( m \). The running time of the algorithm is \( O(m \log^{d-2} m) \) in the worst case and \( O(m) \) in the average case. If \( m \) is equal to \( n^{1/k} \) (\( k > 1 \)), the baseline algorithm is a sublinear time algorithm. If all skyline criteria are independent of each other, the expected error of the algorithm is

\[
\bar{\delta} \leq \frac{n - m}{n} \sum_{i=0}^{d-1} \frac{(\log(m + 1))^i}{i!(m+1)}.
\]

And the standard deviation of the error is \( o(\bar{\delta}) \).

(2). An incremental approximation algorithm is proposed. Given an error bound \( \epsilon \), the algorithm returns an approximate skyline result, whose error is less than the error bound with a probability no less than 99%. The time complexity of the algorithm is \( O(m^{3/2} \log m) \), where \( m \) is the sample size determined by \( \epsilon \) and is almost unaffected by the relation size.

(3). Extensive experiments are performed on three synthetic data sets and a real data set. The synthetic data sets have reached the terabyte level. The experiments verify the theoretical analysis results of the baseline algorithm. The experimental results also show that the incremental algorithm is much faster than the existing skyline algorithms.

The remainder of the paper is organized as follows. Section 2 provides problem definitions. Section 3 describes the baseline algorithm and its analysis. Section 4 presents the incremental approximate algorithm and its analysis. Section 6 concludes the paper.

2 Problem Definition

2.1 Skyline Definition

Let \( T(A_1, A_2, \ldots, A_d) \) be a relation with \( n \) tuples and \( d \) attributes, abbreviated as \( T \). In the following discussions, we assume that all attributes are skyline criteria. First, we formally define dominance relationship between tuples in \( T \).

Definition 2.1. (Dominance between Tuples) Let \( t \) and \( t' \) are in relation \( T \). \( t \) dominates \( t' \) with respect to the \( d \) attributes of \( T \), denoted by \( t < t' \), if \( t.A_i \leq t'.A_i \) for all \( A_i \in \{A_1, \ldots, A_d\} \), and \( \exists A_j \in \{A_1, \ldots, A_d\} \) such that \( t.A_j < t'.A_j \).
Based on the dominance relationship between tuples, we can define the dominance relationship between sets. In the following, \( t \preceq t' \) denotes \( t \prec t' \) or \( t = t' \) with respect to \( d \) attributes of \( T \).

**Definition 2.2. (Dominance between Sets)** A tuple set \( Q \) dominates another tuple set \( Q' \), denoted by \( Q \preceq Q' \), if for each tuple \( t' \) in \( Q' \), there is a tuple \( t \) in \( Q \) such that \( t \prec t' \) or \( t = t' \), i.e. \( t \preceq t' \). \( Q' \) could be one tuple.

Now, we give the definition of skyline of a relation.

**Definition 2.3. (Skyline)** Given a relation \( T(A_1, A_2, ..., A_d) \), the skyline of \( T \) is \( \text{Sky}(T) = \{ t \in T | \forall t' \in T, t' \not\prec t \} \).

**Definition 2.4. (Skyline Problem)** The skyline problem is defined as follows.

Input: a relation \( T(A_1, A_2, ..., A_d) \).

Output: \( \text{Sky}(T) \).

The skyline problem can be equivalently defined as following optimization problem.

**Definition 2.5. (OP-Sky Problem)** OP-Sky problem is defined as follows.

Input: a relation \( T(A_1, A_2, ..., A_d) \).

Output: \( Q \subseteq T \) such that \( |\{ t \in T | Q \preceq \{ t \} \} | \) is maximized and \( \forall t_1, t_2 \in Q, t_1 \neq t_2 \).

The following theorem 2.1 shows that the Skyline Problem is equivalent to the OP-Sky.

**Theorem 2.1.** The skyline of \( T \) is one of the optimal solutions of the problem \( OP_1 \). If there is no duplicate tuples in \( T \), \( \text{Sky}(T) \) is the unique optimal solution.

This paper focus on the approximation algorithms for solving the OP-Sky problem. The error of the approximation algorithms for input relation \( T \) is defined as \( \left| \frac{\mathcal{DN}(\text{Sky}) - \mathcal{DN}(\tilde{\text{Sky}})}{\mathcal{DN}(\text{Sky})} \right| \), where \( \mathcal{DN}(\text{Sky}) \) is the number of tuples in \( T \) dominated by the approximate solution \( \tilde{\text{Sky}} \), and \( \mathcal{DN}(\text{Sky}) \) is the number of tuples in \( T \) dominated by the exact solution \( \text{Sky} \).

If \( \left| \frac{\mathcal{DN}(\text{Sky}) - \mathcal{DN}(\tilde{\text{Sky}})}{\mathcal{DN}(\text{Sky})} \right| \leq \epsilon \), then \( \epsilon \) is called as the error bound of the approximation algorithms.

In the following sections, we will present two approximation algorithms for solving the OP-Sky problem.

### 3 The Baseline Algorithm and Analysis

#### 3.1 The Algorithm

The baseline algorithm first obtains a sample \( S \) of size \( m \) from the input relation \( T \), and then computes the approximate skyline on \( S \). The computation of skyline can call any existing skyline algorithm on \( S \).
Algorithm 1: The Baseline Algorithm

Input: The relation $T(A_1, A_2, \ldots, A_d)$ with $n$ tuples, the sample size $m$
Output: $\tilde{Sky}$, the approximate skyline of $T$

1. $S$ is the sample of $m$ tuples from $T$;
2. return $\text{getSkyline}(S)$. /* getSkyline cab be any exact skyline algorithm */

3.2 Error Analysis of The Baseline Algorithm

To facilitate the error analysis of the baseline algorithm, we assume that the baseline algorithm is based on sampling without replacement. Let $\delta$ be the error of the baseline algorithm, $\bar{\delta}$ be the expected error of the baseline algorithm, and $\sigma^2$ be the variance of the error.

**The Expected Error** We first analyze the expected error of the baseline algorithm, denoted by $\bar{\delta}$. Assume each tuple in $T$ is a $d$-dimensional i.i.d. (independent and identically distributed) random variable.

If the $n$ random variables are continuous, we assume that they have the joint probability distribution function $F(v_1, v_2, \ldots, v_d) = F(\overline{V})$, where $\overline{V} = (v_1, v_2, \ldots, v_d)$. Let $f(v_1, v_2, \ldots, v_d) = f(\overline{V})$ be the joint probability density function of the random variables. Without loss of generality, assume that the range of variables on each attribute is $[0, 1]$, since the domain of any attribute of $T$ can be transformed to $[0, 1]$.

**Theorem 3.1.** If all the $n$ tuple in $T$ are $d$-dimensional i.i.d. continuous random variables with the distribution function $F(\overline{V})$, then the expected error of the baseline algorithm is

$$
\bar{\delta} = \frac{n - m}{n} \int_{[0,1]^d} f(\overline{V})(1 - F(\overline{V}))^m d\overline{V}
$$

where $m$ is the sample size, $f(\overline{V})$ is the density function of the variables, and the range of variables on each attribute is $[0, 1]$.

**Proof.** Due to $\Delta N(Sky(S)) = \Delta N(S) \leq \Delta N(Sky(T)) = n$ where $n$ is the size of the relation $T$, we have

$$
\delta = \frac{|\Delta N(Sky(T)) - \Delta N(Sky(S))|}{\Delta N(Sky(T))} = \frac{n - \Delta N(S)}{n}
$$

Let $X_i$ be a random variable for $1 \leq i \leq n$. $X_i = 0$ if $t_i$ in $T$ is dominated by the sample $S$, otherwise $X_i = 1$. Thus, we have $\Delta N(S) = n - \sum_{i=1}^{n} X_i$ and $\delta = \frac{\sum_{i=1}^{n} X_i}{n}$. By the linearity of expectations, expected error of the baseline algorithm is $
\bar{\delta} = \frac{\sum_{i=1}^{n} E X_i}{n} = \frac{\sum_{i=1}^{n} Pr(X_i = 1)}{n} = Pr(X_i = 1).

$Pr(X_i = 1)$ is the probability that $t_i$ in $T$ is not dominated by $S$. 
Let $Y_i$ be a random variable for $1 \leq i \leq n$. $Y_i = 0$ if $t_i$ in $T$ is picked up into the sample $S$, otherwise $Y_i = 1$. According to the conditional probability formula, we have

$$Pr(X_i = 1) = Pr(Y_i = 0)Pr(X_i = 1|Y_i = 0) + Pr(Y_i = 1)Pr(X_i = 1|Y_i = 1)$$

If $t_i$ is selected into in $S$, then it is dominated by $S$. Therefore, we have $Pr(X_i = 1|Y_i = 0)$ is equal to 0. Based on sampling with replacement, $Pr(Y_i = 1)$ is equal to $(n - m)/n$. In short, we have

$$Pr(X_i = 1|Y_i = 0) = \frac{n - m}{n} Pr(X_i = 1|Y_i = 1)$$

Assume $t_i$ is not selected into $S$. Let $t_i$ have the value $\overrightarrow{V} = (v_1, v_2, ..., v_d)$. Subsequently, for $j_{th}$ tuple $t'_j$ in $S$, $t'_j$ satisfies the distribution $F$ and is independent of $t_i$. It is almost impossible that $t_i$ has a value equal to $t'_j$ on an attribute. The probability of $t'_j \prec t_i$ is $F(\overrightarrow{V})$. In turn, we have $Pr(t'_j \nless t_i|Y_i = 1) = 1 - F(\overrightarrow{V})$.

Because $S$ is a random sample without replacement, all tuples in $S$ are distinct tuples from $T$. All the tuples in $T$ are independently distributed, so are the tuples in $S$. Therefore, the probability that $S$ doesn’t dominate $\{t_i\}$ is

$$Pr(S \nless \{t_i\}|Y_i = 1) = \prod_{j=1}^{m} Pr(t'_j \nless t_i|Y_i = 1) = (1 - F(\overrightarrow{V}))^m$$

In the analysis above, $\overrightarrow{V}$ is regarded as a constant vector. Since that $\overrightarrow{V}$ is a variable vector and has the density function $f(\overrightarrow{V})$, we have

$$Pr(X_i = 1|Y_i = 1) = \int_{[0,1]^d} f(\overrightarrow{V})(1 - F(\overrightarrow{V}))^m d\overrightarrow{V}$$

Thus the probability that $t_i$ is not dominated by $S$ is

$$Pr(X_i = 1) = \frac{n - m}{n} \int_{[0,1]^d} f(\overrightarrow{V})(1 - F(\overrightarrow{V}))^m d\overrightarrow{V}$$

**Corollary 3.1.** If all the $n$ tuple in $T$ are $d$-dimensional i.i.d. continuous random variables, then the expected error of the baseline algorithm is

$$\overline{\delta} = \frac{n - m}{n} \frac{\mu_{m+1,d}}{m + 1}$$

where $m$ is the sample size and $\mu_{m+1,d}$ is the expected skyline size of a set of $m + 1$ $d$-dimensional i.i.d. random variables with the same distribution.

**Proof.** Let $Q$ be a set of $m + 1$ $d$-dimensional i.i.d. random variables with the distribution function $F$, then the expected skyline size of $Q$ is $\mu_{m+1,d} = (m + 1) \int_{[0,1]^d} f(\overrightarrow{V})(1 - F(\overrightarrow{V}))^m d\overrightarrow{V}$. Based on theorem 3.1, we get the corollary. □
If the $n$ random variables are discrete, we assume that they have the joint probability mass function as follows

$$g(v_1, v_2, ..., v_d) = Pr(A_1 = v_1, A_2 = v_2, ..., A_d = v_d)$$

Let $G(v_1, v_2, ..., v_d) = G(V)$ be the probability distribution function of the variables. Assume that $V$ is the set of all tuples in $T$, i.e. all value vectors of the $d$-dimensional variables.

**Theorem 3.2.** If all the $n$ tuple in $T$ are $d$-dimensional i.i.d. discrete random variables with the distribution function $G(V)$, then the expected error of the baseline algorithm is

$$\delta = \frac{n - m}{n} \sum_{V \in V} g(V)(1 - G(V))^m$$

where $m$ is the sample size, $V$ is the set of all value vectors of the $d$-dimensional variables and $g$ is the mass function.

The proof process is basically the same as theorem 3.1 except that duplicate tuples need to be considered.

Based on theorem 3.1 and 3.2, the relation size has almost no effect on the expected error of the baseline algorithm. In most cases, $m$ is much smaller than $n$ such that $(n - m)/n$ approaches to 1.

**Corollary 3.2.** If all the $n$ tuple in $T$ are $d$-dimensional i.i.d. discrete random variables, then the expected error of the baseline algorithm is

$$\delta \leq \frac{n - m \mu_{m+1,d}}{m + 1}$$

where $m$ is the sample size and $\mu_{m+1,d}$ is the expected skyline size of a set of $m + 1$ $d$-dimensional i.i.d. random variables with the same distribution. If there is no duplicate tuples in $T$, the equality of (2) holds.

**Proof.** Let $Q$ be a set of $m + 1$ $d$-dimensional i.i.d. random variables with the distribution function $G$, then the expected skyline size of $Q$ is

$$\mu_{m+1,d} = (m + 1) \sum_{V \in V} g(V)(1 - G(V) + g(V))^m$$

$$\geq (m + 1) \sum_{V \in V} g(V)(1 - G(V))^m$$

the equality of (2) holds if and only if there is no duplicate tuples in $Q$. Based on theorem 3.2, we get the corollary. \qed

By the analysis of the expected skyline size under stronger assumptions in [10], we further analyze the expected error of the baseline algorithm.
**Definition 3.1.** (Component independence) $T(A_1, A_2, \ldots, A_d)$ satisfies component independence (CI), if all $n$ tuples in $T$ follow below conditions.

1. **(Attribute Independence)** the values of tuples in $T$ on a single attribute are statically independent of the values on any other attribute;
2. **(Distinct Values)** $T$ is sparse, i.e. tuples in $T$ have different values on each attribute.

**Theorem 3.3.** Under CI, the error of the baseline algorithm are unaffected by the specific distribution of $T$.

**Proof.** If $T$ satisfies component independence, it can be converted into a uniformly and independently distributed set. After conversion, the error of the baseline algorithm remains unchanged. The specific conversion process is as follows. Consider each attribute in turn. For $A_i$, sort tuples in ascending order by values on $A_i$. Rank 0 is allocated the lowest value 0 on $A_i$, and so forth. Assign the value $j/n$ to $A_i$ of the tuple with rank $j$.

From [10], we have

**Lemma 3.1.** Under CI, the expected skyline size of $T$ is equal to the $(d - 1)^{th}$ order harmonic of $n$, denoted by $H_{d-1,n}$.

For integers $k > 0$ and integers $n > 0$, $H_{d,n} = \sum_{i=1}^{n} \frac{H_{d-1,i}}{i}$. From [6] [8], we have

$$H_{d,n} = \frac{(\log n)^d}{d!} + \gamma \frac{(\log n)^{d-1}}{(d-1)!} + O((\log n)^{d-2}) \leq \sum_{i=0}^{d} \frac{(\log n)^i}{i!}$$

where $\gamma = 0.577...$ is Euler’s constant.

From definition 3.1, there is no duplicate tuples in $T$ under CI. Thus, based on corollary 3.1 and 3.2, we have the following corollary.

**Corollary 3.3.** If the relation $T(A_1, A_2, \ldots, A_d)$ with $n$ tuples satisfies CI, then the expected error of the baseline algorithm is

$$\delta = \frac{n - m}{n(m + 1)} H_{d-1,n} \leq \frac{n - m}{n(m + 1)} \sum_{i=0}^{d-1} \frac{(\log (m + 1))^i}{i!}$$

where $m$ is the sample size.

If only attribute independence condition in definition 3.1 holds and there are tuples in $T$ have the same values on a attribute, we have the following corollary.

**Corollary 3.4.** If all attributes in $T(A_1, A_2, \ldots, A_d)$ are independent of each other, then the expected error of the baseline algorithm is

$$\delta \leq \frac{n - m}{n(m + 1)} H_{d-1,n} \leq \frac{n - m}{n(m + 1)} \sum_{i=0}^{d-1} \frac{(\log (m + 1))^i}{i!}$$

where $m$ is the sample size.
Corollary 3.5. If all attributes in $T(A_1, A_2, ..., A_d)$ are independent of each other, with sample size $m$ equal to $n^{\frac{k}{k+1}} - 1 (k > 1)$, then the expected error of the baseline algorithm is

$$\delta \leq \frac{n - n^{\frac{k}{k+1}} + 1}{n} \sum_{i=0}^{d-1} \frac{(\log n)^i}{k^i n^{\frac{i}{k+1}}!}$$

where $m$ is the sample size.

Variance of The Error. We assume that each tuple in $T$ is a $d$-dimensional i.i.d random variable. Assume that $T$ satisfies component independence (CI). Without losing generality, all random variables are uniformly distributed over $[0, 1]^d$.

Theorem 3.4. If the relation $T(A_1, A_2, ..., A_d)$ with $n$ tuples satisfies CI, then $\sigma^2 = O(\frac{m^{\frac{d}{m-1}}}{n})$, and $\sigma = O(\frac{\sqrt{\log n}}{m}) = o(\delta)$.

Proof. Let $X_i$ be a random variable for $1 \leq i \leq n$. $X_i = 0$ if $t_i$ in $T$ is dominated by the sample $S$, otherwise $X_i = 1$. From the proof in theorem 3.1, we have

$$\sigma^2 = D(\delta) = D(\frac{\sum_{i=1}^{n} X_i}{n}) = D(\sum_{i=1}^{n} X_i)/n^2$$

$$= (E(\sum_{i=1}^{n} X_i^2) + E(\sum_{i \neq j} X_i X_j) - E^2(\sum_{i=1}^{n} X_i))/n^2$$

$$= Pr(X_i = 1)/n + \frac{n-1}{n} Pr_{i \neq j}(X_i = X_j = 1) - Pr^2(X_i = 1).$$

Assume the $t_i$ in $T$ has the value $\mathbf{U} = (u_1, u_2, ..., u_d)$ and the $j$th tuple $t_j$ has the value $\mathbf{V} = (v_1, v_2, ..., v_d)$. Let $(\eta)$ be $\{(\mathbf{U}, \mathbf{V}) | \mathbf{U} \in [0, 1]^d, \mathbf{V} \in [0, 1]^d, u_1 \leq v_1, ..., u_\eta \leq v_\eta, v_{\eta+1} < u_{\eta+1}, ..., v_d < u_d\}.$

$(\eta)$ represents the set of all possible $(\mathbf{U}, \mathbf{V})$, in which $\mathbf{U}$ has values no more than $\mathbf{V}$ on the first $\eta$ attributes and has higher values on the subsequent attributes. Then we have

$$Pr_{i \neq j}(X_i = X_j = 1)$$

$$= \frac{(n-m)(n-1-m)}{n(n-1)} \sum_{\eta=0}^{d} \binom{d}{\eta} \int_{(\eta)} \left(1 - \prod_{i=1}^{d} u_i - \prod_{i=1}^{\eta} v_i + \prod_{i=1}^{\eta} u_i \prod_{i=\eta+1}^{d} v_i\right)^m d\mathbf{U} d\mathbf{V}.$$

In the above equation, $\frac{(n-m)(n-1-m)}{n(n-1)}$ is the probability that two distinct tuples both are not selected into the sample. Based on [1], we have

$$\sum_{\eta=1}^{d-1} \binom{d}{\eta} \int_{(\eta)} \left(1 - \prod_{i=1}^{d} u_i - \prod_{i=1}^{\eta} v_i + \prod_{i=1}^{\eta} u_i \prod_{i=\eta+1}^{d} v_i\right)^m d\mathbf{U} d\mathbf{V} = \frac{\mu_{m+2,d}^2 + O(\mu_{m+2,d})}{(m+1)(m+2)}.$$
Thus,

\[
\Pr_{i \neq j}(X_i = X_j = 1) = \frac{(n-m)(n-1-m)}{n(n-1)} \left( 2 \int_{[0,1]^d} (1 - \prod_{i=1}^d v_i)^m \prod_{i=1}^d v_i dV + \mu_{m+2,d}^2 + O(\mu_{m+2,d}) \right)
\]

\[
= \frac{(n-m)(n-1-m)}{n(n-1)} \left( 2\mu_{m+2,d}(2) + \mu_{m+2,d}^2 + O(\mu_{m+2,d}) \right)
\]

Equation (3) is based on variable substitution. In (4), \( \mu_{n,d}(r) \) denotes the expected size of the \( r \)th layer skyline of \( T \), where the \( r \)th layer skyline of \( T \) is the set of tuples in \( T \) that are dominated by exactly \( r - 1 \) tuples in \( T \), and its expected size is equal to

\[
\mu_{n,d}(r) = n \binom{n-1}{r-1} \int_{[0,1]^d} (1 - \prod_{i=1}^d v_i)^{n-r} \left( \prod_{i=1}^d v_i \right)^{r-1} dV.
\]

Due to \( \Pr(X_i = 1) = \frac{n-m+1}{n-m} \), we have

\[
D \left( \sum_{i=1}^n X_i \right) = (n-m) \mu_{m+1,d} \left( \frac{\mu_{m+2,d}}{m+1} \right) - \frac{(n-m)(n+1)}{(m+1)^2(m+2)} \mu_{m+1,d}^2 + \frac{(n-m)(n-1-m)}{(m+1)(m+2)} \mu_{m+2,d} + O(\mu_{m+2,d})
\]

\[
= (n-m) \mu_{m+1,d} \left( \frac{\mu_{m+2,d}}{m+1} \right) - \frac{(n-m)(n+1)}{(m+1)^2(m+2)} \mu_{m+1,d}^2 + \frac{(n-m)(n-1-m)}{(m+1)(m+2)} \mu_{m+2,d} + O(\mu_{m+2,d}).
\]

Equation (5) holds because \( \mu_{m+2,d} - \mu_{m+1,d} \leq 1 \) and \( \mu_{m+2,d}(2) \leq \mu_{m+2,d} \). With \( \mu_{m+1,d} \leq m+1 \), it is true that \( D \left( \sum_{i=1}^n X_i \right) = O \left( \frac{n^2}{m} \mu_{m,d} \right) \).

### 3.3 Analysis of The Time Complexity

**Theorem 3.5.** If `getSkyline` in step 2 is based on FLET [3], then the time complexity of the baseline algorithm is \( O(m \log^{d-2} m) \) in the worst case, and \( O(m) \) in the average case.

**Proof.** Since the time complexity of FLET [3] is \( O(n \log^{d-2} n) \) in the worst case, and \( O(n) \) in the average case, step 2 of the algorithm need \( O(m \log^{d-2} m) \) time. Thus, the time complexity of the algorithm is \( O(m \log^{d-2} m) \) because that step 1 of the algorithm needs \( O(m) \) time. □
Corollary 3.6. If sample size $m$ equal to $n^{\frac{1}{k}} (k > 1)$, then the running time of the baseline algorithm is $O(n^{\frac{1}{k}} \log^{d-2} n^{\frac{1}{k}})$ in the worst case, and $O(n^{\frac{1}{k}})$ in the average case.

Corollary 3.6 tells that the algorithm is a sublinear time algorithm if the sample size $m$ is equal to $n^{\frac{1}{k}} (k > 1)$.

4 The Incremental Algorithm and Analysis

4.1 The Algorithm

In this section, we devise a sampling-based incremental algorithm, to return an approximate skyline result efficiently with a given error bound, denoted by $\epsilon$. The algorithm first draws an initial sample of size $m_v$, and then computes the approximate skyline on the sample. Afterwards, the algorithm verifies the error of the current approximate result, denoted by $\delta$. If there is no guarantee that $\delta$ isn’t higher than $\epsilon$, the algorithm doubles the sample size and repeats the above process. Otherwise, the algorithm terminates.

The algorithm judges whether $\delta$ meets the requirement by Monte Carlo method. The algorithm obtains a random sample $V$ of size $m_v$, and then counts the total proportion of tuples in $V$ not dominated by the approximate skyline result $\tilde{Sky}$, denoted by $\hat{\delta}$. If $\hat{\delta} \leq \frac{\epsilon}{2}$, the incremental algorithm ensures that the error of $Sky$ is not higher than the error bound.

Algorithm 2: The Incremental Algorithm

\begin{verbatim}
Input: The set of tuples $T$ with size $n$ and $d$ attributes; 
$\epsilon$: the target error bound 
Output: An approximate skyline result 
1 $m_c = m_v = \lceil \frac{2\ln(100)}{\epsilon^2} \rceil$; 
2 $S[1, ..., m_c]$ is the sample of $m_c$ tuples; 
3 $\tilde{Sky} = \text{getSkyline}(S[1, ..., m_c])$; 
4 $S[m_c + 1, ..., m_c + m_v]$ is the sample of $m_v$ tuples for verification; 
5 $\hat{\delta} = \text{verifyError}(\tilde{Sky}, S[m_c + 1, ..., m_c + m_v])$; 
6 While ($\hat{\delta} > \epsilon/2$) Do 
7 $m_c = 2m_c$; 
8 $S[m_c/2 + 1, ..., m_c]$ is the sample of $m_c/2 - m_v$ tuples; 
9 $\tilde{Sky} = \text{mergeSkyline}(\tilde{Sky}, \text{getSkyline}(S[m_c/2 + 1, ..., m_c]))$; 
10 $S[m_c + 1, ..., m_c + m_v]$ is the sample of $m_v$ tuples for verification; 
11 $\hat{\delta} = \text{verifyError}(\tilde{Sky}, S[m_c + 1, ..., m_c + m_v])$; 
12 return $A$;
\end{verbatim}

4.2 Error Analysis of The Incremental Algorithm

From [19], we get the following lemma.
Subroutines 1

1 verifyError (Sky, V)
2 count = 0;
3 for each tuple t in V do
4     if t is not dominated by Sky then
5         count += 1;
6 return count/m_v;

Lemma 4.1. Let \( X_1, \ldots, X_m \) be independent and identically distributed indicator random variables, with \( \mu = E[X_i] \). Then, for \( 0 < \rho < 1 \):

\[
Pr\left( \frac{1}{m} \sum_{i=1}^{m} X_i \leq (1 - \rho)\mu \right) \leq \frac{1}{e^{m\mu\rho^2/2}}.
\]

Based on lemma 4.1, we have the theorem 4.1.

Theorem 4.1. If \( \hat{\delta} \leq \frac{\epsilon}{k} \) (\( k > 1 \)) and \( m_v \geq \frac{2\ln(\frac{1}{k})}{(1 - \frac{1}{k})^2\epsilon^2} \), then the probability that \( \delta \) is less than \( \epsilon \), denoted by \( Pr(\delta < \epsilon) \), is no less than \( 1 - p \).

Proof. Let \( X_i \) be a random variable for \( 1 \leq i \leq m_v \). \( X_i = 0 \) if \( t_i \) in \( V \) is dominated by the approximate result \( \tilde{\text{Sky}} \), otherwise \( X_i = 1 \). Obviously \( \hat{\delta} = \frac{1}{m_v} \sum_{i=1}^{m_v} X_i \) and \( E(\hat{\delta}) = Pr(X_i = 1) \). According to the definition of \( \hat{\delta} \), the probability that a tuple randomly selected from \( T \) is not dominated by \( \tilde{\text{Sky}} \) is \( \frac{n - DN(\tilde{\text{Sky}})}{n} = \delta \), and then \( E(\hat{\delta}) = \delta \).

Based on lemma 4.1 and , let \( \rho \) be equal to \( \frac{(1 - \frac{1}{k})\epsilon}{\delta} \), then we have

\[
Pr(\hat{\delta} + (1 - \frac{1}{k})\epsilon \leq \delta) \leq \frac{1}{e^{m_v(1 - \frac{1}{k})^2\epsilon^2/2}}.
\]

Due to \( \delta \leq 1 \) and \( \hat{\delta} \leq \frac{\epsilon}{k} \), then

\[
Pr(\epsilon \leq \delta) \leq \frac{1}{e^{m_v(1 - \frac{1}{k})^2\epsilon^2/2}}.
\]

If \( m_v \geq \frac{2\ln(\frac{1}{k})}{(1 - \frac{1}{k})^2\epsilon^2} \), then \( \frac{1}{e^{m_v(1 - \frac{1}{k})^2\epsilon^2/2}} \) is no more than \( p \), and the same for \( Pr(\epsilon \leq \delta) \).

\( \Box \)

Corollary 4.1. The error of the approximate skyline result returned by the incremental algorithm is less than the error bound with a probability no less than 99%.
4.3 Analysis of Time Complexity

Theorem 4.2. If getSkyline and mergeSkyline are based on FLET [3], then the time complexity of the incremental algorithm is $O(m^3 \log m)$, where $m$ is the final sample size and is almost a constant relative to the relation size.

Proof. The final sample size of the incremental algorithm is $m = m_c + m_v$, where $m_c$ is the final value at the end of the algorithm and is not less than $m_v$.

If $m_c$ is equal to $m_v$, then $m_c = m_v = m/2 = O(\frac{1}{\epsilon})$ and the algorithm calls getSkyline once and verifyError once. Since the time complexity of FLET [3] is $O(n \log^{d-2} n)$, the running time of getSkyline is $O(m \log^{d-2} m)$. Based on corollary 3.1 and 3.2, the size of $\tilde{Sky}$ is at most $\epsilon m = O(\frac{1}{\epsilon}) = O(m^2)$, and then the running time of verifyError is $O(\epsilon m n_v) = O(\frac{1}{\epsilon}) = O(m^2)$. Therefore, the current time complexity of the algorithm is $O(\frac{1}{\epsilon}) = O(m^2)$, which is unaffected by the relation size.

If $m_c$ is greater than $m_v$, then $m_c = O(m) > m_v = O(\frac{1}{\epsilon})$. The calculation process of the algorithm is completely equivalent to SD&C in [4]. Then the total running time of getSkyline and mergeSkyline is $T_1(m, d) = 2T_1(m/2, d) + M(m, d)$, where $M(m, d)$ is equal to $O(m \log^{d-2} m)$. Then we have $T_1(m, d) = O(m \log^{d-1} m)$. The times of verifications is at most $\log(m/m_v) = O(\log(m \epsilon^2))$.

Based on corollary 3.1 and 3.2, the size of $\tilde{Sky}$ is at most $O(\epsilon m)$. Therefore, due to $\epsilon m_v = O(\frac{1}{\epsilon}) = O(m^2)$, the total running time of verifyError is $O(\epsilon m n_v \log(m^2)) = O(m^2 \log m)$. Thus the current time complexity of the algorithm is $O(m^2 \log m)$. Based on theorem 3.1 and 3.2 at this time, $m$ is almost unaffected by the relation size. \qed

5 EXPERIMENTAL RESULTS

5.1 Experimental Settings

We implemented the approximation algorithm in C++, and then ran the algorithm on Dell OptiPlex-7500(4 Cores, 8 Threads 3.6GHz i7 CPU + 16G memory + 64 bit Linux). The experimental results are computed by averaging 20 executions of the approximation algorithm.

Extensive experiments are performed on three synthetic data sets (independent distribution, correlated distribution and anti-correlated distribution) and a real data set. Each tuple has 8 numeric attributes. For independent distribution, the values of tuples on each attribute are generated uniformly and independently. For correlated distribution, the values of tuples on the first two attributes are generated with Pearson Correlation Coefficient (PCC) 0.5. For anti-correlated distribution, the values of tuples on the first two attributes are generated with PCC $-0.5$. The real data set comes from UCI Machine Learning Repository [9], and are kinematic properties measured by the particle detectors in the accelerator.
5.2 Experiment 1: The Effect of Data Volume on Error

In experiment 1, we evaluate the effect of data volume on the error of the approximation algorithm, with a sample of size 10000 and 2 skyline criteria. The data volumes considered are 0.2TB, 0.4TB, 0.6TB, 0.8TB and 1.0TB. The experiment is performed on the three synthetic data sets. As shown in figure 1(a), the error of the approximation algorithm is unaffected by the relation size for any distribution, which is a significant advantage of the algorithm.

5.3 Experiment 2: The Effect of Skyline Criteria Size on Error

In experiment 2, we evaluate the effect of the skyline criterion size $d$ on the error of the approximation algorithm, with a sample of size 10000. The skyline criterion sizes considered are 2, 3, 4, 5. The experiment is performed on all four data sets. The considered data volumes of the three synthetic data sets are 0.2TB, and that of the real data set is 10GB. As shown in figure 1(b), for all data sets, as the dimensionality increases, the error of the approximation algorithm significantly grows.
5.4 Experiment 3: Predicted and Real Error

In experiment 3, we verify the connection between the expected skyline cardinality and the expected representative ratio, with 2 skyline criteria. The sample sizes considered are 0.1K, 1K, 10K, 100K and 1000K (K=10^3). The experiment is performed on all four data sets. The considered data volumes of the three synthetic data sets are 0.2TB, and that of the real data set is 10GB. We can use the skyline proportion of the sample to approximate \( \frac{m+\sqrt{d}}{m+1} \), and then use it to predict the expected error of the approximation algorithm, based on corollary 3.1 and 3.2. As shown in figure 1(c) for each data set, the curves of the predicted and real errors closely fit. The fact that the predicted and real representative ratios are nearly equal shows the correctness of corollary 3.1 and 3.2.

5.5 Experiment 4: Standard Deviation of Error

In experiment 4, we evaluate the variance of the algorithm error, with varying sample sizes and 2 skyline criteria. Basic experiment settings in experiment 4 are basically the same as experiment 3. As shown in figure 1(d) for all data sets, as the sample size increases, the standard deviation decreases significantly. If the sample size is equal to 10K, the standard deviation is less than 0.001 for all data sets. With a moderate size sample, the standard deviation is relatively small.

6 Conclusion

In this paper, we proposed two sampling-based approximation algorithms for processing skyline queries on big data. The first algorithm draws a random sample of size \( m \) and computes the approximate skyline on the sample. The expected error of the algorithm is almost independent of the input relation size and the standard deviation of the error is relatively small. The running time of the algorithm is \( O(m \log d - 2m) \) in the worst case and \( O(m) \) in the average case. Experiments show that with a moderate size sample, the algorithm has a low enough error. The second algorithm returns an approximate skyline result with a given error bound \( \epsilon \). The time complexity of the algorithm is \( O(m^2 \log m) \), where \( m \) is the sample size determined by \( \epsilon \) and is almost unaffected by the relation size. Experiments show that the algorithm is much faster than the existing skyline algorithms.

References

1. Bai, Z.D., Chao, C.C., Hwang, H.K., Liang, W.Q.: On the variance of the number of maxima in random vectors and its applications. In: Advances In Statistics, pp. 164–173. World Scientific (2008)
2. Bartolini, I., Ciaccia, P., Patella, M.: Efficient sort-based skyline evaluation. ACM Transactions on Database Systems (TODS) 33(4), 1–49 (2008)
3. Bentley, J.L., Clarkson, K.L., Levine, D.B.: Fast linear expected-time algorithms for computing maxima and convex hulls. Algorithmica 9(2), 168–183 (1993)
4. Bentley, J.L., Kung, H.T., Schkolnick, M., Thompson, C.D.: On the average number of maxima in a set of vectors and applications. Journal of the ACM (JACM) 25(4), 536–543 (1978)
5. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings 17th international conference on data engineering. pp. 421–430. IEEE (2001)
6. Buchta, C.: On the average number of maxima in a set of vectors. Information Processing Letters 33(2), 63–65 (1989)
7. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: ICDE. vol. 3, pp. 717–719 (2003)
8. Devroye, L.: A note on finding convex hulls via maximal vectors. Information Processing Letters 11(1), 53–56 (1980)
9. Frank, A.: Uci machine learning repository. http://archive.ics.uci.edu/ml (2010)
10. Godfrey, P.: Skyline cardinality for relational processing. In: International Symposium on Foundations of Information and Knowledge Systems. pp. 78–97. Springer (2004)
11. Godfrey, P., Shipley, R., Gryz, J., et al.: Maximal vector computation in large data sets. In: VLDB. vol. 5, pp. 229–240 (2005)
12. Han, X., Li, J., Yang, D., Wang, J.: Efficient skyline computation on big data. IEEE Transactions on Knowledge and Data Engineering 25(11), 2521–2535 (2012)
13. Koltun, V., Papadimitriou, C.H.: Approximately dominating representatives. In: International Conference on Database Theory. pp. 204–214. Springer (2005)
14. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: VLDB’02: Proceedings of the 28th International Conference on Very Large Databases. pp. 275–286. Elsevier (2002)
15. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. Journal of the ACM (JACM) 22(4), 469–476 (1975)
16. Lee, K.C., Lee, W.C., Zheng, B., Li, H., Tian, Y.: Z-sky: an efficient skyline query processing framework based on z-order. The VLDB Journal 19(3), 333–362 (2010)
17. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting stars: The k most representative skyline operator. In: 2007 IEEE 23rd International Conference on Data Engineering. pp. 86–95. IEEE (2007)
18. Magnani, M., Assent, I., Mortensen, M.L.: Taking the big picture: representative skylines based on significance and diversity. The VLDB journal 23(5), 795–815 (2014)
19. Mitzenmacher, M., Upfal, E.: Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis. Cambridge university press (2017)
20. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data. pp. 467–478 (2003)
21. Soholm, M., Chester, S., Assent, I.: Maximum coverage representative skyline. In: EDBT. pp. 702–703 (2016)
22. Tan, K.L., Eng, P.K., Ooi, B.C., et al.: Efficient progressive skyline computation. In: VLDB. vol. 1, pp. 301–310 (2001)
23. Tao, Y., Ding, L., Lin, X., Pei, J.: Distance-based representative skyline. In: 2009 IEEE 25th International Conference on Data Engineering. pp. 892–903. IEEE (2009)
24. Xingxing Xiao, J.L.: Sampling based approximate skyline calculation on big data (2020). https://arxiv.org/abs/2008.08103