Kernel-Based Skyline Cardinality Estimation

Zhenjie Zhang[1], Yin Yang[2], Ruichu Cai[3], Dimitris Papadias[2], Anthony Tung[1]

[1]Department of Computer Science
National University of Singapore
Computing 1, Singapore, 117590
{zhenjie, atung}@comp.nus.edu.sg

[2]Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{yini, dimitris}@cse.ust.hk

[3]School of Computer Science and Engineering
South China University of Technology
Guangzhou, China
cairuichu@scut.edu.cn

ABSTRACT

The skyline of a *d*-dimensional dataset consists of all points not dominated by others. The incorporation of the skyline operator into practical database systems necessitates an efficient and effective cardinality estimation module. However, existing theoretical work on this problem is limited to the case where all *d* dimensions are independent of each other, which rarely holds for real datasets. The state of the art Log Sampling (LS) technique simply applies theoretical results for independent dimensions to non-independent data *anyway*, sometimes leading to large estimation errors. To solve this problem, we propose a novel Kernel-Based (KB) approach that approximates the skyline cardinality with *nonparametric* methods. Extensive experiments with various real datasets demonstrate that KB achieves high accuracy, even in cases where LS fails. At the same time, despite its numerical nature, the efficiency of KB is comparable to that of LS. Furthermore, we extend both LS and KB to the *k*-dominant skyline, which is commonly used instead of the conventional skyline for high-dimensional data.

Categories and Subject Descriptors

H.2 DATABASE MANAGEMENT, H.2.4 Systems—Query processing, H.2.8 Database applications

General Terms

Algorithms, Performance, Experimentation, Theory

Keywords

Skyline, Cardinality Estimation, Kernel, Non-Parametric Methods

1. INTRODUCTION

Given a *d*-dimensional dataset *DS*, we say that a point $p \in DS$ *dominates* another $q \in DS$, if and only if *p* is better than *q* on at least one dimension, and no worse on any of the remaining *d*–1 dimensions. The skyline $SKY_{DS} \subseteq DS$ is the set of points not dominated by others in *DS*. Skyline queries are particularly useful for selecting records according to multiple, sometimes contradicting, criteria [10][15]. Consider the *MobilePhone* table shown in Figure 1 containing 5 records m_1-m_5. Each phone is associated with a *price* and *standby* time attribute, and can be thought of as a point in the 2-dimensional (price-standby) space. Clearly a low price and long standby time are desirable properties for any user. Given these preferences, the skyline contains m_1, m_3, and m_5; m_2 is dominated by m_3 since it is more expensive and has shorter standby life. Similarly, m_4 is dominated by m_5. Note that although m_5 has neither the longest standby time nor the lowest price, it is in the skyline as a more balanced choice. In general, the skyline consists of all records that are potentially the best choice for any user, regardless of the relative weight of different attributes.

In practical applications, a skyline query often involves other relational operators, e.g., selection conditions on the records to be retrieved [27], and projections that focus on a subspace of all attributes [38]. Moreover, complex queries may combine skylines with joins. Assume, for instance, that the shop owner wants to identify the customer IDs who purchased models that belong to the skyline of *MobilePhone*. In the example database of Figure 1, customers c_1, c_3 satisfy the requirement as they bought skyline models m_1 and m_3. Processing such a query involves computing the skyline of *MobilePhone*, as well as joining *MobilePhone* with *Purchase*. An execution plan could perform the join before the skyline, while another could reverse the order of the two operators. To choose the most efficient plan in an educated manner, the DBMS should effectively estimate the output size of the skyline. This paper focuses on the problem of skyline cardinality estimation (SCE), which also plays a pivotal role in the cost analysis of the skyline operator [9][15].

MobilePhone

mid	standby	price
m1	200	200
m2	300	400
m3	400	350
m4	250	320
m5	300	300

Customer

cid	name
c1	James
c2	Lily
c3	George

Purchase

oid	mid	cid	date
o1	*m3*	*c1*	5/21/08
o2	*m2*	*c2*	5/21/08
o3	*m1*	*c1*	5/22/08
o4	*m4*	*c2*	5/22/08
o5	*m3*	*c3*	5/22/08

Figure 1 Example skyline application

SIGMOD'09, June 29–July 2, 2009, Providence, RI, USA.

The current state-of-the-art for SCE is Log Sampling (LS) [9], implemented in Microsoft SQL Server. LS assumes that the skyline cardinality m, of an arbitrary dataset of size n, follows the model $m = A\log^B n$ for some constants A, B. Although this property holds for data with *independent* dimensions, it is not true in many practical situations. In particular, we show analytically and experimentally that two large classes of datasets violate the property. The first involves anti-correlated attributes, i.e., if a record has a good value in one attribute (e.g., long standby life), it is likely to have a "bad" value on the remaining ones (e.g., have a high price). The second type of datasets that lead to failure of LS are those consisting of multiple clusters (e.g., *MobilePhone* may contain two clusters in the price-standby plane corresponding to PDAs and music phones).

We solve these problems through a novel Kernel-Based approach (KB). Unlike LS, KB is *nonparametric*, meaning that it does not rely on any assumptions about the relationship between the skyline cardinality m and the dataset size n, and, thus, is generally more robust. Extensive experiments confirm that KB achieves low estimation error in a variety of real and synthetic datasets, and is significantly more accurate than LS. Moreover, the efficiency of KB is comparable to that of LS. As a second step, we extend both LS and KB to the k-dominant skyline [8], which is more meaningful than the conventional skyline for high-dimensional datasets.

The rest of the paper is organized as follows. Section 2 surveys related work and necessary background for kernel-based statistics. Section 3 describes the theoretical foundations and the algorithmic framework of KB. Section 4 adapts LS and KB to k-dominant skylines. Section 5 contains an extensive experimental evaluation. Finally, Section 6 concludes the paper with directions for future work.

2. RELATED WORK

Section 2.1 overviews the literature on the skyline query and its variants. Section 2.2 surveys skyline cardinality estimation, focusing on LS. Section 2.3 presents kernel-based methods.

2.1 The Skyline Query

The skyline query, also known as the Maximal Vector Problem [3], was first studied in the computational geometry literature. Early work focuses on main-memory solutions. The Double Divide-and-Conquer (DD&C) algorithm [23] achieves the lowest worst-case cost among all known methods. In particular, DD&C answers a skyline query in $O(n\log^{d-2} n)$ time, where n is the data cardinality and $d\geq 2$ the dimensionality. Subsequent work, such as Linear Divide-and-Conquer [3] and Fast Linear Expected-Time [2], use DD&C as a sub-routine and, thus, preserve its worst-case bound, while improving its average-case performance. As pointed out in [15], these solutions are highly theoretical in nature, and commonly incur practical drawbacks, such as large constants, poor scalability for large d, and considerable implementation overhead.

Börzsönyi et al. [5] introduced the skyline problem to the database community proposing two disk-based solutions, based on the divide-and-conquer (D&C) and block-nested loop (BNL) paradigms. SFS [11] improves BNL by sorting all tuples on one of the d dimensions in a pre-processing step. The pre-sorting idea is enhanced in LESS [15], ZSearch [24] and SaLSa [1]. When the data dimensionality is relatively low, a multi-dimensional index can speed up processing [22][27][34]. Several papers study incremental maintenance of skylines in the presence of frequent updates, common in data stream environments [25][35][37].

Another line of research deals with alternative forms of skyline processing. Chan et al. [7] and Sacharidis et al. [32] study skylines in partially-ordered domains (e.g., a user may prefer "Nokia" to "Motorola", but considers "Nokia" and "LG" incomparable). Pei et al. [28] compute probabilistic skylines in uncertain datasets (e.g., a model may have 300 hours stand-by time with probability 80%, and 200 hours with probability 20%). Given that the skyline of high-dimensional data is usually very large, the k-dominant skyline [8] and approximately-dominating skyline [21] relax the definition of dominance to obtain more meaningful results. SubSky [36] returns the skyline in a subspace containing only selected attributes. SkyCube [38] pre-computes the skyline in all subspaces, and BestView [29] finds the skyline in the subspaces that are most meaningful semantically. Finally, [26] accelerates skyline processing in low-cardinality domains, e.g., *MobilePhone* may include a binary attribute indicating whether or not the model has Bluetooth capabilities.

2.2 Skyline Cardinality Estimation

Most selectivity estimation work focuses on conventional queries such as ranges and joins. Common techniques include sampling (e.g., [17]), histograms (e.g., [16][20][30]), kernels (e.g., [4][16]), etc. However, it is hard to apply these methods to skyline cardinality estimation (SCE), due to the special characteristics of the skyline operator. First, the geometric shape of the skyline, which is required in many histogram and kernel based methods, is not known in advance without fully computing it. Second, the skyline cardinality does not grow linearly with the sample size, and it is non-trivial to compute it from the local skyline of the sample set. Furthermore, skyline cardinality varies significantly depending on the data distribution. Figure 2 illustrates four typical 2D distributions: independent, correlated, anti-correlated, and clustered. Correlated (resp. anti-correlated) datasets have fewer (resp. more) skyline points than independent ones[1]. Clustered datasets require complex models that aggregate the cardinality in individual clusters.

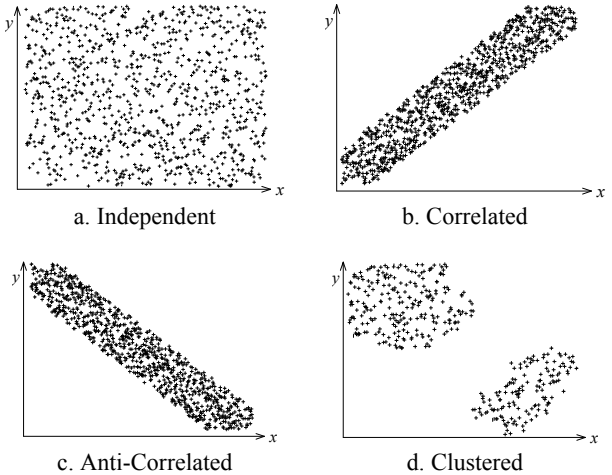


Figure 2 Four common data distributions

[1] For this example and the rest of the presentation, we assume that smaller values are preferable (i.e., the skyline points have a small distance to the beginning of the axes).

Given the dataset cardinality n, Bentley et al. [3] prove that when all d dimensions are independent, the expected skyline cardinality m equals the (d−1)-*th order harmonic* $H_{d-1,n}$ of n, defined recursively as follows:

$$H_{0,n} = 1$$
$$H_{j,n} = \sum_{i=1}^{n} \frac{H_{j-1,i}}{i} \quad 1 \le j \le d-1 \tag{1}$$

The original proof is limited to the case where no two points share the same value on any dimension. Godfrey [14] removes this restriction and shows that for sufficiently large n: $m = H_{d-1,n} \approx \ln^{d-1} n / d!$, where $d!$ denotes the factorial of d. Chaudhuri et al. [9] generalize the formula $\ln^{d-1} n / d!$ to $A\log^B n$ (where A, B are constants) in order to capture other (than independent) datasets. The main justification is that for datasets (e.g., anti-correlated), whose skyline cardinality grows faster than independent ones, B is set to a value larger than d−1, and vice versa. Based on this model, Log Sampling (LS) draws a random sample S from the dataset DS, splits it into two parts S_1 and S_2 containing $|S_1|$, $|S_2|$ points respectively, and computes their skyline cardinalities $|SKY_{S_1}|$, $|SKY_{S_2}|$. Since S_1, S_2 have the same distribution as DS, their skyline cardinalities are expected to follow the same model as that of DS. Therefore, from equations $|SKY_{S_1}| = A\log^B |S_1|$ and $|SKY_{S_2}| = A\log^B |S_2|$, LS solves constants A and B as follows:

$$B = \frac{\log|SKY_{S_2}| - \log|SKY_{S_1}|}{\log(\log|S_2|) - \log(\log|S_1|)}, A = \frac{|SKY_{S_1}|}{\log^B |S_1|} \tag{2}$$

$A\log^B n$ is used as an estimate for the global skyline cardinality m. Although LS is rather accurate on small, synthetic datasets [9], we argue that it has two fundamental shortcomings. First, many datasets simply do not follow the $A\log^B n$ model, independently of the values of A and B. For example, in anti-correlated data (Figure 2c), it is common that m grows *linearly* with n, i.e., m=Cn for a constant C. No matter how large B is, Cn always grows faster than $A\log^B n$. Consequently, the error of LS increases with n, and can be arbitrarily large, according to the following equation:

$$\forall A \forall B, \lim_{n\to\infty}(Cn - A\log^B n) = +\infty \tag{3}$$

The second problem is that even for datasets whose skyline cardinality *does* follow the $A\log^B n$ model, the sample size of LS may have to be unrealistically large to capture the correct value for B. Consider that the local skyline cardinalities of the two clusters in Figure 2d follow models $A_1\log^{B_1} n$ and $A_2\log^{B_2} n$ respectively, with $B_1 \neq B_2$. Since points from different clusters do not dominate each other, the global skyline contains $m = A_1\log^{B_1} n_1 + A_2\log^{B_2} n_2$ points. Without loss of generality, assume $B_1 > B_2$; then, the first term gradually dominates with increasing n leading to $m = \Theta(\log^{B_1} n)$. Yet, the influence of the second term diminishes only when n is sufficiently large, as $A_1\log^{B_1} n$ grows sub-linearly. Using a smaller sample set, Equation (2) is likely to yield a value for B that is between B_1 and B_2, and hence, leads to unbounded error when applied to estimate the global skyline cardinality.

2.3 Kernel Density Estimation

Nonparametric methods, such as kernels [18], radial basis function [19] and exploratory projection pursuit [13], are widely used to estimate the probability density function (PDF) of a dataset from a random sample. Their main advantage is that they are not based on any assumptions on the underlying data, and, thus, are generally more robust than parametric techniques such as LS. We focus on kernel methods as they are most commonly used in practice. Figure 3 illustrates 2D kernel density estimation (KDE) with a sample set of 3 points (–2, 0), (0, 1) and (3, 2). KDE initializes the PDF for the positions holding sample points to 1, and the remaining ones to 0. Then, it associates a *kernel* with each of the three samples, which spreads the unit PDF of the sample to the entire domain according to a specific *kernel function*.

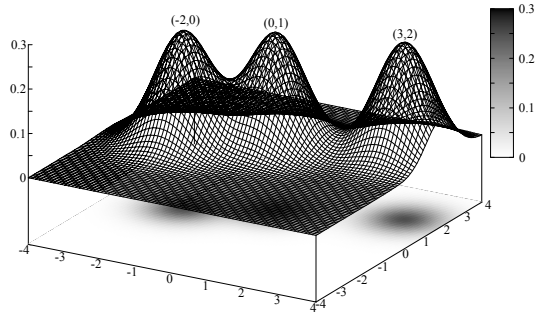


Figure 3 KDE with three sample points

In the following, we use the terms "sample" and "kernel" interchangeably. Most kernel functions assign positions closer to the sample point a higher portion of the unit PDF. Intuitively, one can imagine each kernel as a "light source", illuminating the initially "dark" domain space. The final PDF value for each position is calculated by summing up the influence of all kernels upon it. For example, in Figure 3, the PDF of positions lying between (–2, 0) and (0, 1) are boosted by both kernels, and, thus, is higher compared to that of other points close to only one kernel, e.g., those around (3, 2). Formally, let S, $|S|$, d be a random sample set, its cardinality, and dimensionality respectively. The PDF of an arbitrary position q is given by the following equation:

$$PDF(q) = \sum_{s\in S}\left(\frac{1}{|S|h^d\sqrt{det(\Sigma)}} K\left(\frac{dist_\Sigma(q,s)}{h}\right)\right) \tag{4}$$

where h (a real value) and Σ (a $d\times d$ matrix) are tunable parameters, called the *kernel bandwidth* and *kernel orientation* respectively; $det(\Sigma)$ denotes the determinant of matrix Σ; $dist_\Sigma(p, s)$ signifies the *Mahalanobis distance* [18] between q and s. As a special case, when Σ is an identity matrix, $dist_\Sigma(q, s)$ equals the Euclidean distance between q and s. Note that Equation (4) uses the same value of h and Σ for all kernels (referred to as *fixed kernels* in the literature). Another option is to have individual values h_s and Σ_s for each kernel s (called *adaptive kernels*), which replace all occurrences of h and Σ in the above formula. K is the *kernel function*; a popular choice for K is the Gaussian Kernel [18], defined by the following equation:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{5}$$

Figure 4 visualizes the kernel bandwidth and orientation with three Gaussian kernels s_1-s_3. Both s_1 and s_2 use the identity matrix for the orientation Σ, with s_1 having a larger bandwidth than s_2. Intuitively, the unit PDF of s_1 is more evenly spread out over the entire space, while that of s_2 is more concentrated around the sample point. Kernel s_3 has a different orientation; its shape

resembles a rotated oval rather than a circle, reflecting the distortion Σ imposes on the distance function $dist_\Sigma$.

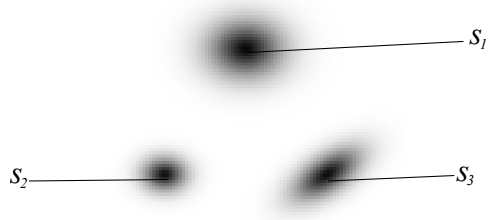


Figure 4 Kernels with different bandwidths and orientations

3. KB SKYLINE CARDINALITY ESTIMATION

Section 3.1 discusses the scaling invariance property of skylines. Section 3.2 describes the main concepts of KB, and Section 3.3 focuses on the computation of the kernel bandwidth. Section 3.4 presents an alternative implementation of KB based on adaptive kernels. Table 1 summarizes frequent symbols used throughout the section.

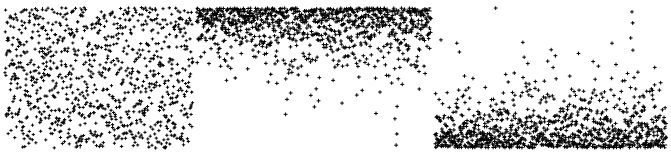
Table 1 Frequent symbols

Symbol	Meaning
DS	Input dataset for skyline cardinality estimation
d	Number of dimensions of *DS*
n	Number of points in *DS*
m	Estimated number of skyline points in *DS*
S	Random sample set of *DS*
SKY_S	Sample skyline set of *S*
IDR(*p*)	Inverse dominance region of point *p*
PDF(*p*)	Probability density function at position *p*
Ω_p	Cumulated probability density for IDR_p
h, Σ	Kernel bandwidth and orientation (for fixed kernels)
erf(*x*)	Error function [31] (defined in Equation 10)
p_{LB}[i]	Lower bound of dimension *i* of the domain
p_{UB}[i]	Upper bound of dimension *i* of the domain
S_p	Subset of samples in *S* within distance 2*h* from point *p*

3.1 Scaling-Invariance of the Skyline

A very important property of the skyline is *scaling invariance*. Specifically, any dataset *DS* has the same skyline cardinality as its discretized counterpart *DS′*, obtained as follows. Let $p[i]$, $1 \le i \le d$ be the value of point *p* on the *i*-th dimension. Each $p \in DS$ is mapped to a point $p' \in DS'$, such that $p'[i]$ $(1 \le i \le d)$ is the *rank* of $p[i]$ when all points in *DS* are sorted in increasing order of their *i*-th dimension. For example, if $DS = \{p_1=(-2, 0), p_2=(0, 1), p_3=(3, 2)\}$, we have $p'_1 = (1, 1)$, since p_1 has the smallest co-ordinates on both dimensions. Similarly, $p'_2 = (2, 2)$, and $p'_3 = (3, 3)$. It can be proven [15] that if *p* is in the skyline of *DS*, then p' is in the skyline of *DS′*, and vice versa.

Scaling invariance leads to the equivalence of datasets with very different distributions, with respect to the skyline cardinality. Consider the three 2D datasets DS_1, DS_2 and DS_3 shown in Figure 5. In DS_1, values are uniformly distributed in the range (0, 1). DS_2 is obtained by applying the following transformation on DS_1: for each point $(x, y) \in DS_1$, there is a corresponding one $(x', y') \in DS_2$, such that $x' = x$ and $y' = \log y$. Similarly, DS_3 is generated by creating for each $(x, y) \in DS_1$, its counterpart $(x'', y'') \in DS_2$, satisfying $x'' = x$ and $y'' = -\log(1-y)$. Since both transformations are *order-preserving*, the results of discretization for all three datasets are the same, and, thus, they have identical skyline cardinality. On the other hand, their PDF functions are entirely different. This leads to the conclusion that SCE and density estimation are distinct problems. This issue is further discussed in Section 3.3.



a. Dataset DS_1 b. Dataset DS_2 c. Dataset DS_3

Figure 5 Datasets with identical skyline cardinality

3.2 Main Concepts of KB

Given a dataset *DS*, KB first draws a random sample *S* and computes the skyline SKY_S of *S*. To distinguish the skyline sets SKY_S and SKY_{DS} of the sample *S* and the entire dataset *DS* respectively, we refer to the former as the *sample skyline*, and the latter as the *global skyline*. The cardinality of *DS*, SKY_{DS}, *S* and SKY_S are denoted as *n*, *m*, $|S|$ and $|SKY_S|$, respectively. Let Ψ_{DS} be the actual probability that a random data point in *DS* belongs to the global skyline; the expected value for *m* can be obtained as:

$$m = n \cdot \Psi_{DS} \tag{6}$$

Accordingly, SCE reduces to the problem of estimating Ψ_{DS}. It is worth clarifying that Equation (6) does *not* imply a linear relationship between *m* and *n*. In particular, Ψ_{DS} is strictly associated with the global skyline, and the probability Ψ_S for a sample point in *S* to lie on the sample skyline can differ from Ψ_{DS}, i.e., $\Psi_{DS} \neq \Psi_S$, where $\Psi_S = |SKY_S|/|S|$. Figure 6 illustrates a sample *S* of 5 points p_1-p_5, whose sample skyline SKY_S consists of p_1, p_4 and p_5. Clearly, the probability for a sample point to lie on SKY_S is 3/5 = 0.6. On the other hand, regarding the global skyline, p_2 and p_3 are definitely not in SKY_{DS}, while p_1, p_4, p_5 *may* be in SKY_{DS}, if they are not dominated by other data points not in *S*. The example shows that the estimation of Ψ_{DS} entails assessing the probability for each sample skyline point to be dominated by a tuple in $DS \neg S$.

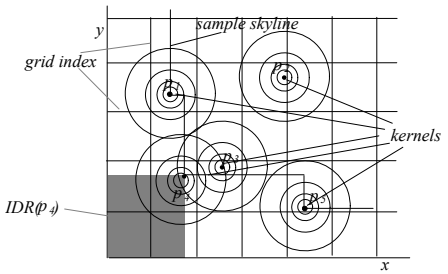


Figure 6 Kernels and local skyline points

We define the *inverse dominance region* (IDR) of a point *p* as:

$$IDR(p) = \{q \mid q[i] \le p[i], \quad \forall 1 \le i \le d\} \tag{7}$$

where $p[i]$, $q[i]$ denote the value on the *i*-th dimension of points *p* and *q*, respectively. Observe that *q* is not restricted to be a point in *DS*. In the example of Figure 6, the IDR of p_4 is the shaded area. Let Ω_p be the probability that a random point of *DS* falls in *IDR*(*p*); the probability for a sample skyline point $p \in SKY_S$ to reside on the global skyline is thus $(1-\Omega_p)^{n-|S|}$. Accordingly, Ψ_{DS} is estimated as:

$$\Psi_{DS} \approx \Psi_{DS}^{EST} = \frac{1}{|S|} \sum_{p \in SKY_S} \left(1 - \Omega_p\right)^{n-|S|} \quad (8)$$

KB approximates Ω_p with kernels. Specifically, it associates each sample point in S with a kernel. These kernels are used to evaluate the probability density function (PDF) for all positions in the d-dimensional space. We apply kernels with fixed bandwidth h and identity matrix for the orientation Σ (alternative models are discussed in Section 3.4). Note that although each such kernel models the local data assuming independent and symmetric dimensions, the combination of multiple kernels can capture *arbitrary* data distributions[2]. Figure 7 illustrates examples of multiple symmetric kernels. The kernels in Figure 7a (resp. Figure 7b) correspond to correlated (resp. clustered) data. Symmetric kernels are commonly used in practice [12] because more complicated kernels do not necessarily lead to better results, while the may incur additional performance overhead [18].

a. Correlated b. Clustered

Figure 7 Combination of symmetric kernels

Replacing the kernel function K and distance *dist* with their respective definitions, we transform Equation (4) as follows:

$$PDF(q) = \frac{1}{|S|h^d} \sum_{s \in S} \frac{1}{\sqrt{2\pi}} e^{-\sum_{i=1}^{d} (s[i]-q[i])^2 \Big/ 2h^2} \quad (9)$$

Ω_p is then calculated by the cumulated probability of $IDR(p)$, i.e., the integration of $PDF(q)$ over all positions in $IDR(p)$. Formally:

$$\begin{aligned}
\Omega_p &= \int_{IDR(p)} PDF(q)\,dq \\
&= \frac{1}{|S|h^d} \sum_{s \in S} \int_{IDR(p)} \frac{1}{\sqrt{2\pi}} e^{-\sum_{i=1}^{d} (s[i]-q[i])^2 / 2h^2} dq \\
&= \frac{1}{|S|h^d} \sum_{s \in S} \prod_{i=1}^{d} \int_{-\infty}^{p[i]} \frac{1}{\sqrt{2\pi}} e^{-(s[i]-q[i])^2/2h^2} dq[i] \\
&= \frac{1}{2|S|h^d} \sum_{s \in S} \prod_{i=1}^{d} \left(1 + erf\left(\frac{s[i]-p[i]}{h\sqrt{2}}\right)\right) \\
\text{where } erf(x) &= \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt
\end{aligned} \quad (10)$$

The first step in the above derivation simply replaces $PDF(q)$ with its concrete definition of Equation (9). The second step exploits the facts that (i) $IDR(p)$ is an axis-parallel hyper-rectangle and (ii) $PDF(q)$ is symmetric for all d dimensions, and, thus, the integral is independent in all d dimensions. The function *erf* in the last step is the *error function* [31]. Although *erf* does not have a closed form solution, it can be approximated very accurately (within error 10^{-12} for any input) and efficiently (involving only one exponential evaluation and a small number of multiplications) with Chebyshev fitting [31], which is implemented in most scientific libraries, e.g., GSL (www.gnu.org/software/gsl/). Equations (6)-(10) then jointly estimate the global skyline cardinality m.

KB applies two additional optimizations in the computation of Ω_p, which significantly boost accuracy and efficiency, respectively. The first one addresses the issue that data points in DS are usually restricted in a finite domain D, defined by a pair of points (p_{LB}, p_{UB}) such that for each dimension i ($1 \le i \le d$), the value $p[i]$ of every point $p \in DS$ is bounded by $p_{LB}[i]$ and $p_{UB}[i]$, i.e., $p_{LB}[i] \le p[i] \le p_{UB}[i]$. In this situation we need to apply two changes to the computation of Ω_p. First, the inverse dominance region (IDR) is redefined to take D into account, as follows:

$$IDR_D(p) = \{q \mid p_{LB}[i] \le q[i] \le p[i], \quad \forall 1 \le i \le d\} \quad (11)$$

Correspondingly, Ω_p is computed by integrating the PDF over $IDR_D(p)$ instead of $IDR(p)$. Furthermore, we need to address the *boundary effect* [4] of the kernels. Simply stated, the problem is that since a kernel stretches throughout the *unbounded* d-dimensional data space, some of its influence is *outside* the domain D. Consequently, the PDF values for positions inside D are smaller than what they should be. We use a simple remedy that normalizes the total PDF in D to 1 for each kernel [4]. The resulting Ω_p is presented in Equation (12). We omit a detailed derivation due to space limitations.

$$\begin{aligned}
\Omega_p &= \frac{1}{2|S|h^d} \sum_{s \in S} \prod_{i=1}^{d} \frac{(\varphi(p) - \varphi(p_{LB}))}{(\varphi(p_{UB}) - \varphi(p_{LB}))} \\
\text{where } \varphi(p) &= 1 + erf\left(\frac{s[i]-p[i]}{h\sqrt{2}}\right)
\end{aligned} \quad (12)$$

The second optimization exploits the fact that when a kernel s is sufficiently far away from a sample skyline p, the influence of s on p is negligible. Specifically, for Gaussian kernels, 95% of a kernel's unit PDF is distributed to points within distance $2h$ from its center. Accordingly, KB uses $2h$ as a threshold, and reduces unnecessary computations by only considering the subset S_p of kernels whose minimum distance (denoted by $MinDist$) to $IDR(p)$ is no larger than $2h$. In the example of Figure 6, if p_1, p_2, p_5 are farther away than $2h$ from IDR(p_4), then $S_{p_4} = \{p_3, p_4\}$. Applying this idea, Equation (12) is transformed to:

$$\begin{aligned}
\Omega_p &= \frac{1}{2|S|h^d} \sum_{s \in S_p} \prod_{i=1}^{d} \frac{(\varphi(p) - \varphi(p_{LB}))}{(\varphi(p_{UB}) - \varphi(p_{LB}))} \\
\text{where } S_p &= \{s \mid s \in S \wedge MinDist(s, IDR(p)) \le 2h\} \\
\text{and } \varphi(p) &= 1 + erf\left(\frac{s[i]-p[i]}{h\sqrt{2}}\right)
\end{aligned} \quad (13)$$

To accelerate the retrieval of S_p for each sample skyline point p, KB pre-processes the sample points with a d-dimensional grid. Each cell has length $2h$ on every dimension, which ensures that only those cells that intersect with, or are adjacent to, $IDR(p)$ may possibly contain kernels with minimum distance $2h$ to $IDR(p)$, and, thus, have non-negligible influence on Ω_p. Note that since p is in the local skyline, cells fully contained in $IDR(p)$ must be empty. In the example of Figure 6, $IDR(p_4)$ intersects with 3 cells (excluding the covered cell at the low-left corner), and is adjacent to another 5. From these cells, KB only retrieves p_3 and p_4, and avoids checking other sample points, which are guaranteed not to

[2] Likewise, histograms summarize arbitrary distributions, even though they assume uniform distribution within each bucket.

be in S_p. The grid index is effective when the dimensionality d is relatively low. Its effect gradually decreases as d grows due to the *curse of dimensionality*. Figure 8 summarizes *KB*. The bandwidth h controls the scope of each kernel and its setting affects significantly the accuracy of *KB*. Next, we focus on the computation of h.

KB(DS, S, SKY_S, h)
// INPUT = DS: the entire dataset
// S, SKY_S: a random sample of DS, and the skyline of S
// h: the fixed kernel bandwidth
// OUTPUT = m: estimated global skyline cardinality
1. Organize all sample points in a grid index of cell length $2h$
2. For each point p in SKY_S
3. Initialize S_p to empty set
4. For each cell C that intersects with, or is adjacent to $IDR(p)$
5. For each sample point $s \in C$
6. If $MinDist(s, IDR(p)) \le 2h$, add s to S_p
7. Compute Ω_p according to Equation (13)
8. Compute m according to Equations (6) and (8), and return m

Figure 8 *KB* algorithm

3.3 Kernel Bandwidth Computation

In the kernel density estimation literature, a popular choice for kernel bandwidth is the optimal value h_{MISE} that minimizes the *mean integrated squared error* (MISE) [18]. Specifically, for Gaussian kernels, h_{MISE} is calculated as follows:

$$h_{MISE} = \left(\frac{4}{(2d+1)|S|} \right)^{\frac{1}{d+4}} \tag{14}$$

In SCE, however, there does not exist a single best value for h based on d and $|S|$ alone. This is due to the scaling-invariance property of skylines, which imposes equivalence for datasets with completely different distributions. We illustrate this fact with the example of Figure 5. Recall that all three datasets share exactly the same skyline cardinality. If we draw sample sets from DS_2 and DS_3, their skyline cardinalities are also expected to be the same, assuming that samples follow their respective global distributions. However, in DS_2 the skyline points generally fall into sparse regions, while those in DS_3 are in dense areas. Therefore, if KB uses the same kernel bandwidth (e.g., h_{MISE}) for both datasets, the IDR of a sample skyline in DS_2 (resp. DS_3) which lies in a sparse (resp. dense) region, is far away from (resp. closer to) its surrounding kernels, and, thus receives rather weak (resp. strong) influence from them. Consequently, the calculated probability Ω_p for a sample skyline point to be dominated by others is far smaller in DS_2 than in DS_3. Therefore, KB would produce a much larger estimate for DS_2 than for DS_3. Since the two datasets have the same skyline cardinality, at least one of the estimates incurs high error.

Without a universal formula, KB resorts to a validation based approach to compute the appropriate kernel bandwidth. Specifically, it draws two random sample sets S and S', referred to as the *training set* and *validation set* respectively, such that $S \cap S' = \varnothing$. Then, it computes the local skyline $SKY_{S'}$ of S', and obtains its cardinality $|SKY_{S'}|$. With the training set S and a given value for the bandwidth h (but without S'), KB is able to compute an estimate $|SKY_{S'}|^{EST}$, as described below. The goal of tuning h is that the difference between $|SKY_{S'}|$ and $|SKY_{S'}|^{EST}$ is no larger than $\theta \cdot |S'|$, where $|S'|$ is the cardinality of S', and θ is a pre-defined threshold. Specifically, let $\Psi_{S'} = |SKY_{S'}| \,/\, |S'|$ denote the probability for a random point in S' to be in the skyline of S'. Its estimation from the training set is:

$$\Psi_{S'} \approx \Psi_{S'}^{EST} = \frac{1}{|S|} \sum_{p \in SKY_S} \left(1 - \Omega_p\right)^{|S'|} \tag{15}$$

Comparing Equations (8) and (15), the latter simply substitutes n-$|S|$ with $|S'|$. The target h must satisfy $|\Psi_{S'} - \Psi_{S'}^{EST}| \le \theta$. To compute it, KB initially sets h to h_{MISE} defined in Equation (14), and calculates $\Psi_{S'}^{EST}$ with the procedure described in Section 3.2. If the goal is achieved, h is considered satisfactory and the estimated value for the global skyline cardinality is returned; otherwise, h has to be tuned further.

The tuning of h follows the *Newton-Raphson* framework, an iterative process that usually converges very fast: if the final value for h has b bits, the expected number of iterations is O(log b) [31]. In addition, *Newton-Raphson* is robust, in the sense that even if it does not reach the best h, it still converges to a local minimum. Specifically, let h_{old} (h_{new}) be the bandwidth value before (after) tuning, respectively; h_{new} is calculated with the following equation:

$$h_{new} = h_{old} - \frac{\Psi_{S'} - \Psi_{S'}^{EST}}{\frac{d\Psi_{S'}^{EST}}{dh}(h_{old})} \tag{16}$$

where $d\Psi_{S'}^{EST}/dh$ is the first-order derivative of $\Psi_{S'}^{EST}$ as a function of h. We next compute this derivative, assuming that KB uses Equation (10) to obtain Ω_p. The case in which KB applies Equations (12) or (13) leads to longer formulae for $d\Psi_{S'}^{EST}/dh$, but the computations are based on the same principles. In preparation of deriving $d\Psi_{S'}^{EST}/dh$, we first obtain $dPDF(q)/dh$, which is straightforward following the definition of $PDF(q)$ and basic rules of derivatives:

$$\begin{aligned} dPDF(q)/dh &= d\left(\sum_{s \in S} \frac{1}{|S|h^d} K\left(\frac{dist(q,s)}{h} \right) \right) \Big/ dh \\ &= \sum_{s \in S} \left(\frac{-d}{|S|h^{d+1}} K\left(\frac{dist(q,s)}{h} \right) + \frac{1}{|S|h^d} dK\left(\frac{dist(q,s)}{h} \right) \Big/ dh \right) \\ \text{where } dK\left(\frac{dist(q,s)}{h} \right) \Big/ dh &= d\left(\frac{1}{\sqrt{2\pi}} e^{-dist^2(q,s)/2h^2} \right) \Big/ dh \\ &= \frac{2dist^2(q,s)}{\sqrt{2\pi}\,2h^3} e^{-dist^2(q,s)/2h^2} = \frac{dist^2(q,s)}{h^3} K\left(\frac{dist(q,s)}{h} \right) \end{aligned} \tag{17}$$

Then, we apply the results of Equation (17) in (18):

$$\begin{aligned} d\Psi_{S'}^{EST}/dh &= d\left(\frac{1}{|S|} \sum_{p \in SKY_S} \left(1 - \Omega_p\right)^{|S'|} \right) \Big/ dh \\ &= \frac{-|S'|}{|S|} \sum_{p \in SKY_S} \left(1 - \Omega_p\right)^{|S'|-1} \times d\Omega_p/dh \\ \text{where} \\ d\Omega_p/dh &= d\left(\int_{IDR(p)} PDF(q)dq \right) \Big/ dh = \int_{IDR(p)} \left(dPDF(q)/dh \right) dq \\ &= \int_{IDR(p)} \sum_{s \in S} \left(\frac{-d}{|S|h^{d+1}} K\left(\frac{dist(q,s)}{h} \right) + \frac{dist^2(q,s)}{|S|h^{d+3}} K\left(\frac{dist(q,s)}{h} \right) \right) dq \\ &= \frac{-d}{h}\Omega_p + \sum_{s \in S} \int_{IDR(p)} \frac{dist^2(q,s)}{|S|h^{d+3}} K\left(\frac{dist(q,s)}{h} \right) dq \end{aligned} \tag{18}$$

Finally, using $Y_i = -(s[i]-q[i])^2/2h^2$, we obtain:

$$\int_{IDR(p)} \frac{dist^2(q,s)}{h^3} K\left(\frac{dist(q,s)}{h}\right) dq = \int_{IDR(p)} \frac{-2\sum_{i=1}^{d} Y_i}{\sqrt{2\pi}h} e^{\sum_{j=1}^{d} Y_j} dq$$
$$= \frac{-2}{\sqrt{2\pi}h} \sum_{i=1}^{d} \int_{IDR(p)} Y_i \prod_{j=1}^{d} e^{Y_i} dq$$
$$= \frac{-2}{\sqrt{2\pi}h} \sum_{i=1}^{d} \int_{-\infty}^{p[i]} Y_i e^{Y_i} dq[i] \prod_{j=1, j\neq i}^{d} \int_{-\infty}^{p[j]} e^{Y_j} dq[j] \quad (19)$$
$$= \frac{-1}{\sqrt{2\pi}h} \sum_{i=1}^{d} \left(1+\gamma\left(\tfrac{3}{2}, Y_i\right)\right) \prod_{j=1, j\neq i}^{d} \left(1+erf\left(s[i]-p[i] \Big/ h\sqrt{2\pi}\right)\right)$$
$$\text{where } \gamma(a,x) = \int_0^x t^{a-1}e^{-t}dt,\ erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

Similar to Equation (10), the derivation in Equation (19) exploits the fact that the integrand can be decomposed into the product of d independent components, and that the region $IDR(p)$ is an axis-parallel hyper-rectangle. In the last step of Equation (19), $\gamma(a, x)$ is the *incomplete gamma function* [31], which, like *erf*, can be approximated accurately and efficiently. We summarize the bandwidth computation algorithm in Figure 9.

Compute_Bandwidth (*DS*, *S*, SKY_S, θ)
// INPUT = *DS*: the entire dataset
// *S*, SKY_S: a random sample of *DS*, and the skyline of *S*
// τ: pre-defined thresholds controlling output accuracy
// OUTPUT = *h*: kernel bandwidth
1. Initialize *h* to h_{MISE}, defined by Equation (14)
2. Draw a sample *S′* from the *DS*, such that $S \cap S' = \varnothing$
3. Compute the local skyline $SKY_{S'}$ of *S′*, and $\Psi_{S'} = |SKY_{S'}|/|S'|$
4. Repeat
5. Compute $\Psi_{S'}^{EST}$ with Equation (15) and the *KB* algorithm
6. If $|\Psi_S - \Psi_{S'}^{EST}| > \theta$, replace *h* with h_{new} computed from Equations (16)-(19)
7. Until $|\Psi_S - \Psi_{S'}^{EST}| \leq \theta$
8. Return *h*

Figure 9 Bandwidth computation algorithm

3.4 Discussion

So far we have used kernels with fixed bandwidth *h* and identify-matrix orientation Σ in KB. In the sequel, we first address adaptive bandwidth, and then complex orientations. Consider that each kernel *s* has a distinct h_s. Besides replacing *h* with h_s in all formulae, we also need to calculate $|S|$ different bandwidths. Toward this, KB first computes the initial values $h_{s,0}$ for each *s* using local statistics surrounding *s*, following the standard procedure described in [18]. Then, we assume that there exists a constant δ such that the best value h_s^* of h_s for every *s* satisfies $h_s^* = \delta h_{s,0}$, and apply the *Newton-Raphson* procedure to find δ. In our experiments, the accuracy gains brought by adaptive bandwidths are usually rather small, if not negligible. Similar observations have been made in traditional kernel density estimation [18].

KB can also be easily extended to diagonal (but not necessarily identity) orientation matrices Σ. In this case, the distance can be decomposed into the weighted sum of individual dimensions, i.e., $dist_\Sigma^2(q,s) = \sum_{i=1}^{d} w_i (q[i]-s[i])^2$, where the weights are determined by the diagonal values. Such an orientation Σ is calculated with the global data variations on each dimension [18]. Furthermore, similar to the bandwidth, the orientation can also be *adaptive*, i.e., each kernel *s* may use a distinct Σ_s, computed based on local statistics around *s*. In our experiments, we found that diagonal-matrix orientations indeed improve accuracy for specially designed datasets. For real data, however, they do not have a clear advantage.

Using more complex (i.e., non-diagonal) orientations in KB is much harder. Intuitively, such kernels (e.g., s_3 in Figure 4) can be thought of as "rotated" counterparts of those with diagonal-matrix orientations. Hence, the distance function $dist_\Sigma(q, s)$ can no longer be decomposed into the product of *d* independent, axis-parallel components, invalidating the derivations in Equations (10) and (19). Consequently, the evaluation of the probability Ω_p and its derivative can no longer be performed using special functions such as *erf*. Instead, KB must resort to general-purpose, multi-variant integration approximation techniques (e.g., Monte-Carlo [31]). Such methods are usually expensive or inaccurate, defeating the purpose of using complex orientations.

4. CARDINALITY ESTIMATION FOR *K*-DOMINANT SKYLINES

Besides conventional SCE, the proposed KB framework can be adapted to handle other skyline variants. We focus on the *k-dominant skyline* [8], which is particularly useful for high-dimensional data. According to the definition of the conventional skyline, as *d* grows, it becomes increasingly difficult for a point to dominate another on all dimensions. Consequently, for high dimensional data, the skyline includes a considerable portion of the entire dataset, and may become meaningless. The *k*-dominant skyline avoids this problem by altering the definition of dominance, as follows: a *d*-dimensional point *p* is said to *k-dominate* another *q*, if and only if *p* is better than *q* on at least one dimension and not worse on at least *k* ($k < d$) dimensions. Clearly, if *p* dominates *q*, *p* also *k*-dominates *q* for any *k*, but the reverse does not hold. Revisiting the example of Figure 1, records m_1 and m_2 in the *MobilePhone* table are incomparable according to the conventional dominance definition, since m_1 has a lower price than m_2, while the latter has longer standby time. With respect to 1-dominance, however, the two models dominate each other, since each is preferable on one attribute. In general, transitivity does not hold, and mutual or circular *k*-dominance is common, e.g., between m_1 and m_2.

The set of points not *k*-dominated by others form the *k-dominant skyline*. The *k*-dominant skyline is always a subset of the conventional skyline, usually containing the most meaningful points. However, as there is no direct relationship between the cardinality of the conventional and *k*-dominant skylines, the problem of *k*-dominant skyline cardinality estimation (*k*-SCE) is non-trivial. In Section 4.1, we adapt LS for *k*-SCE. Then, in Section 4.2, we propose *k*-KB, which is based on kernels.

4.1 *k*-LS

To relate the problems of SCE and *k*-SCE, we observe that any point *p* in the *k*-dominant skyline $kSKY_{DS}$ must be a conventional skyline point in every *k*-dimensional subspace $DS|_k$, constructed by projecting the entire dataset *DS* onto *k* out of the *d* attributes. Conversely, if *p* belongs to the skyline of all *k*-dimensional subspaces, it is in $kSKY_{DS}$ as well. Therefore, the *k*-dominant skyline is equivalent to the intersection of conventional skylines in all *k*-dimensional subspaces. Formally:

$$kSKY_{DS} = \bigcap_{\forall DS|_k} SKY_{DS|_k} \quad (20)$$

where $DS|_k$ is any k-dimensional projection of DS

For instance, consider 1-dominance in the *MobilePhone* table. The 1D skyline on attributes *standby* and *price* is $\{m_3\}$ and $\{m_1\}$ respectively. Since these two sets do not overlap, the 1-dominant skyline is empty. As another example, the 2-dominant skyline of a 4-dimensional (say, x, y, z, w) dataset is the intersection of the skylines in the 6 subspaces xy, xz, xw, yz, yw, zw.

Based on the LS methodology, the skyline cardinality of each k-dimensional subspace follows the $m=A\log^B n$ model, with distinct values for A and B in different subspaces. Let B^* be the minimum value of B among all such subspaces, DS^* be the particular subspace having B^*, and A^* be the corresponding value for constant A. Following Equation 20, we have:

$$|kSKY_{DS}| \le |SKY_{DS^*}| = A^* \log^{B^*} n \quad (21)$$

The equality in the above formula holds when all points on SKY_{DS^*} are simultaneously skyline points in every other k-dimensional subspace. On the other hand, $|kSKY_{DS}|$ can be as low as zero, e.g., for the 1-dominant skyline in the example of Figure 1. We thus hypothesize that there exists a constant A' ($0 \le A' \le A^*$) such that $|kSKY_{DS}| = A'\log^{B^*} n$, and use this assumption in kLS. This choice is based on our experimental evaluation, which indicates that LS usually underestimates the true cardinality of conventional skylines. Thus, using the upper-bound for $|kSKY_{DS}|$ compensates for this bias. The computation of A' and B^* can be performed in two different ways, both using two disjoint random sample sets S_1 and S_2 of DS. The first estimates the value of B for every k-dimensional subspace, takes their minimum as B^*, and solves A' from equations $|kSKY_{S_1}| = A'\log^{B^*} |S_1|$ and $|kSKY_{S_2}| = A'\log^{B^*} |S_2|$. The second approach directly solves A', B^* from these two equations. While the second method is faster, it carries the risk of underestimating B^*, and, consequently, may yield larger error for $|kSKY_{DS}|$.

4.2 *k*-KB

Similar to KB, k-KB draws a random sample $S \subset DS$, and computes the sample k-dominant skyline $kSKY_S$. Again we reduce the estimation of the cardinality of the global k-dominant skyline $kSKY_{DS}$ to that of Pr_{kSKY}, i.e., the probability for a random data point to lie in $kSKY_{DS}$. Pr_{kSKY} is in turn reduced to the computation of Ω_p for each sample k-dominant skyline point $p \in kSKY_S$. The first significant difference between KB and k-KB concerns the definition of inverse dominance region (Equation 7 for KB). The corresponding concept used in k-KB is the *k-dominant IDR* (kIDR), defined as follows.

$$kIDR(p) = \{q \mid \exists 1 \le i_1 < i_2 < \ldots < i_k \le d, q[i_j] \le p[i_j], 1 \le j \le k\} \quad (22)$$

We elaborate on the shape of kIDR (k=2) for a 3D point p. Let p_x, p_y, p_z be the value of p on the x, y, z axes respectively. Figure 10 illustrates three axis-parallel hyper rectangles: $\{q|q_y<p_y, q_z<p_z\}$, $\{q|q_x<p_x, q_z<p_z\}$, $\{q|q_x<p_x, q_y<p_y\}$. Each rectangle is bounded in two dimensions by the respective projection of p, and spans over the entire domain in the remaining one. For instance, the leftmost rectangle in Figure 10 is bounded by (p_y, p_z) on y, z axes, and covers the whole x dimension. Any point inside $\{q|q_y<p_y, q_z<p_z\}$ dominates p in the subspace consisting of the y and z axes. Similarly, a point in $\{q|q_x<p_x, q_z<p_z\}$ (middle rectangle in Figure 10) dominates p in the subspace consisting of x and z dimensions. Point p is not 2-dominated, and therefore, it belongs to 2-SKY_{DS}, if all three rectangles are empty. Consequently, 2-$IDR(p)$ is the union of the three rectangles.

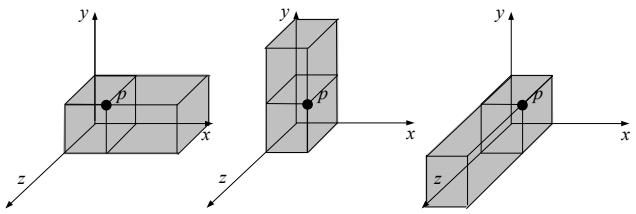


Figure 10 2-*IDR* of a 3D point *p*

In general, the $kIDR$ of a d-dimensional point p is the union of $\binom{d}{k}$ hyper-rectangles, each bounded in k dimensions and unbounded in $(d–k)$ dimensions. Since $kIDR(p)$ has a complex shape, the derivation in Equation (10) is not directly applicable. In order to obtain Ω_p, k-KB (i) decomposes $kIDR(p)$ into $\binom{d}{k}$ *disjoint* partitions, (ii) evaluates the integration of $PDF(q)$ over all positions q in each partition separately, and (iii) sums up the results. Figure 11 shows the decomposition of 2-$IDR(p)$ in Figure 10. The first partition is $P_1=\{q|q_y<p_y, q_z<p_z\}$, i.e., the entire hyper-rectangle bounded on y and z, and unbounded on x. The second one is $P_2=\{q|q_x<p_x, q_z<p_z\}–P_1$, i.e., the hyper-rectangle bounded on x and z and unbounded on y, except for the part already contained in P_1 (to render the two partitions disjoint). The third partition is $P_3=\{q|q_x<p_x, q_y<p_y\}–P_1–P_2$.

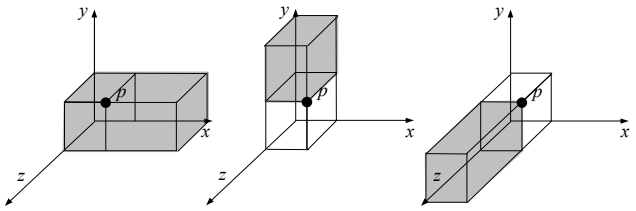


Figure 11 Decomposition of 2-*IDR*(*p*)

Figure 12 describes the general algorithm for decomposing a kIDR. The ith partition is obtained by taking the ith hyper-rectangle forming $kIDR(p)$, and removing parts already covered by previous partitions (Lines 4-6). Because in every dimension, each hyper-rectangle is either unbounded, or bounded at one end by the corresponding value of p, a partition P_i ($1 \le i \le \binom{d}{k}$) returned by *Decompose* is also a d-dimensional, axis-parallel hyper-rectangle. Thus, the derivation of Equation (10) applies to the integration of $PDF(p)$ over every partition P_i ($1 \le i \le \binom{d}{k}$).

Decompose ($kIDR(p)$)
// INPUT = the kIDR of a point p
// OUTPUT = $\binom{d}{k}$ disjoint partitions of $kIDR(p)$
1. Initialize the set $\boldsymbol{P}$ to empty
2. Let $kIDR(p)$ be the union of a set $\boldsymbol{R}$ of $\binom{d}{k}$ hyper-rectangles
3. For i = 1 To $\binom{d}{k}$
4. Initialize $P_i = R_i$, where R_i is the ith hyper-rectangle in $\boldsymbol{R}$
5. For j = 1 To i–1
6. $P_i = P_i - P_j$
7. Add P_i to $\boldsymbol{P}$
8. Return P

Figure 12 Algorithm for decomposing *kIDR*(*p*)

If $\boldsymbol{P} = \{P_i | 1 \le i \le \binom{d}{k}\}$, Ω_p is computed using Equation (23):

$$\Omega_p = \int_{kIDR(p)} PDF(q)\,dq = \sum_{P \in \boldsymbol{P}} \int_P PDF(q)\,dq \qquad (23)$$

The calculation of the best bandwidth h is performed in a similar fashion as in KB, with the additional complication that Ω_p now is the sum of $\binom{d}{k}$ integrals. When the dimensionality d is high, the number of partitions $\binom{d}{k}$ may become very large. Consequently, the evaluation of Equation (23) can be very expensive. For such cases, we propose *adjustable k-KB* (A*k*-KB), which provides a tradeoff between efficiency and accuracy. Specifically, A*k*-KB lets the user specify a parameter τ ($0 \le \tau \le 1$), and estimates Ω_p as follows.

$$\Omega_p = \frac{\sum_{R \in \boldsymbol{R}} volume(R)}{\sum_{P \in \boldsymbol{P}} volume(P)} \sum_{R \in \boldsymbol{R}} \int_R PDF(q)\,dq$$
$$\text{where } \boldsymbol{R} \subset \boldsymbol{P}, \sum_{R \in \boldsymbol{R}} volume(R) \ge \tau \sum_{P \in \boldsymbol{P}} volume(P) \qquad (24)$$

Intuitively, Equation (24) considers only a subset $\boldsymbol{R}$ of the partitions $\boldsymbol{P}$, whose combined volume in the d-dimensional space is no smaller than τ times the total volume of all partitions in $\boldsymbol{P}$, and extrapolates the result obtained from $\boldsymbol{R}$. To minimize the number of partitions evaluated, A*k*-KB sorts the partitions of $\boldsymbol{P}$ in decreasing order of their volumes, and selects the first $|\boldsymbol{R}|$ partitions that satisfy the above condition. Clearly, a smaller value of τ leads to a faster estimate, while a larger τ achieves higher accuracy and stability.

Finally, KB can be used for cardinality estimation in other types of queries related to skylines. For instance, given a set of points Q, the *spatial skyline* retrieves the objects that are the nearest neighbors of any point in Q [33]. In this case, a data point p dominates another p', if p is closer than p' to every query point. Although spatial skylines necessitate specialized query processing algorithms, in terms of SCE the problem can be thought of as $|Q|$-dimensional skyline, where each dimension corresponds to the distance from a query point. Thus, KB is directly applicable, and as evaluated in our experiments, rather accurate.

5. EXPERIMENTAL EVALUATION

We implemented LS and KB, as well as their adaptations for the *k*-dominant skyline, in C++. LS is based exactly on the description of [9]. KB uses fixed-bandwidth kernels with identity orientation. All experiments are performed on a Pentium 4 2.4G CPU, using the following datasets:

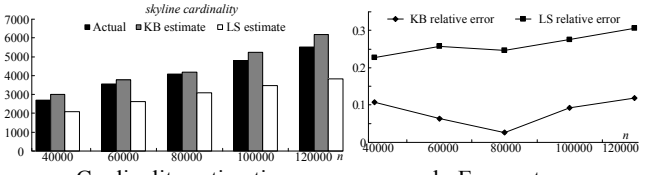
- *Household* (available at *www.ipums.org*) contains 127K tuples. Each record has 6 attributes that store the percentage of an American family's annual income spent on: gas, electricity, water, heating, insurance, and property tax. Smaller values are considered better for the skyline query.
- *Corel* (available at *kdd.ics.uci.edu*) includes data about 68K images. Each image is encoded in the HSV (i.e., hue, saturation and value) space, and the 9 dimensions correspond to the mean, standard deviation and skewness of the image's pixels in the H, S, V channels. The skyline prefers lower values for the mean, and standard deviation, and higher values for skewness.
- *NBA* (available at *www.basketballreference.com*) contains data about 20K basketball players in 300K matches. Each player is associated with 16 different statistics, e.g., number of games played, minutes played, total points, etc. Larger values are preferred for all attributes. Following most previous work (e.g., [28]), we consider appearances of the same player in different matches as distinct records. The skyline contains the set of outstanding (i.e., not dominated) performance records over all matches.
- *Spatial skyline* (available at *www.rtreeportal.org*) has the 2D co-ordinates of 1.25 million points in Los Angeles. A set Q of query points are randomly chosen from this set. For every remaining point, we measure its distance to each query point as the $|Q|$ dimensions to be considered in the skyline query, preferring smaller values. According to [33], only query points that reside on the convex hull of Q affect the result of the skyline query. Hence, we manually ensure that all points in Q are on the convex hull to eliminate this effect.
- *Synthetic datasets*, with independent (abbreviated as IND), correlated (COR) and anti-correlated (ANT) dimensions, created[3] with the dataset generator of [5].

In experiments where we need to alter the cardinality n of a real dataset, we randomly select a fraction of the records. The maximum value of n corresponds to the (original) cardinality of the dataset. Section 5.1 reports the results for conventional SCE. Section 5.2 deals with *k*-dominant skylines.

5.1 Conventional Skyline

The initial set of experiments evaluates the accuracy of SCE as a function of the dataset cardinality n. For each setting, we execute KB and LS with 10 different random sample sets (i.e., obtained with different random seeds), and report the median results. Because KB involves more complex computations, and, thus, higher running time, we compensate LS by giving it a larger sample size. Specifically, in all experiments involving n, KB uses two disjoint sample sets S and S' (see Section 3), whose sizes are fixed to 500 and 1000, whereas LS uses 3000 samples for S_1 and another 6000 for S_2 (Section 2.2). As we show later, with these sample sizes, KB and LS have comparable CPU overhead.

We first evaluate KB and LS on the *household* dataset. Figure 13a plots the absolute values of the actual skyline cardinality m, as well as the estimates m_{KB} and m_{LS} produced by KB and LS. Figure 13b shows the relative error rates ε_{KB} and ε_{LS} of the two methods, calculated as $\varepsilon_{KB} = |m_{KB} - m| / m$ and $\varepsilon_{LS} = |m_{LS} - m| / m$, respectively. In all settings, LS leads to higher error despite its larger sample size. Notably, when n = 40000, LS samples over 20% of the entire dataset, and yet incurs a considerable error of 23%. Furthermore, ε_{LS} increases with n, whereas ε_{KB} is affected mainly by random fluctuations. An interesting observation is that KB and LS are complementary, in the sense that m_{KB} tends to overestimate m, while m_{LS} usually underestimates it.



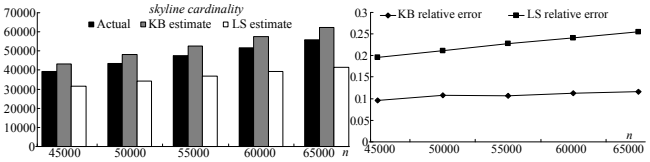
a. Cardinality estimations b. Error rates

Figure 13 Accuracy vs. dataset cardinality (*household*)

Figure 14 repeats the same experiments on the *Corel* dataset. *Corel* differs from *household* in that the former has a much larger

[3] Correlated and anti-correlated datasets are produced by rotating independent ones.

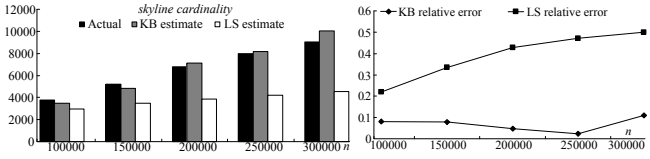
skyline, containing a considerable fraction of the whole dataset. Nevertheless, the observations that (i) KB consistently outperforms LS, and (ii) the relative error rate of LS steadily increases with n, still hold. Moreover, ε_{KB} is more stable in *Corel* than in *household*, indicating that *Corel* contains fewer outliers, leading to better density estimates using kernels.



a. Cardinality estimations b. Error rates

Figure 14 Accuracy vs. dataset size n (*Corel*)

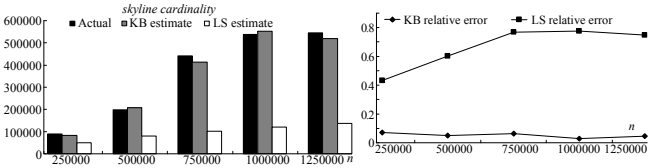
Figure 15 reports results on *NBA*. Compared with the previous two datasets, *NBA* has the following distinct features: (i) more records (300K) (ii) larger dimensionality (16), and (iii) higher correlation between attributes, since a (good) player who has long playing time is likely to perform more activities (e.g., score points, take rebounds). Again, KB outperforms LS in all settings, and the performance gap expands fast with the data cardinality. Specifically, when n is 10^6, the performance of LS is comparable to that on *household* and *Corel*. On the other hand, when n reaches 3 million, ε_{LS} is around 50%. KB fits this dataset well, yielding error around 10%.



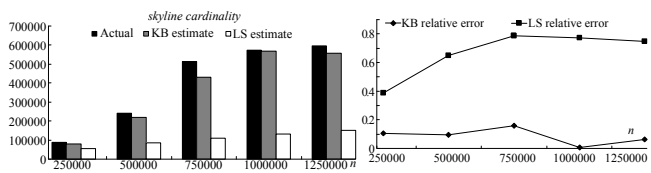
a. Cardinality estimations b. Error rates

Figure 15 Accuracy vs. dataset size n (*NBA*)

Figure 16 compares KB and LS on *spatial skylines*. Since *Corel* and *NBA* cover high-dimensional data, we focus on low-dimensional queries, using 4 and 5 query points (abbreviated as *4QP* and *5QP*). The sizes of these datasets (up to 1.25 million) are also much larger than those in the previous experiments. The estimates of KB are always very close to the actual skyline cardinality. LS, however, suffers from severe underestimation and, for large n, ε_{LS} reaches 80%. Note that the dataset (points in Los Angeles) underlying *spatial skyline* contains clusters that correspond to densely populated areas, which partially explains the poor performance of LS.



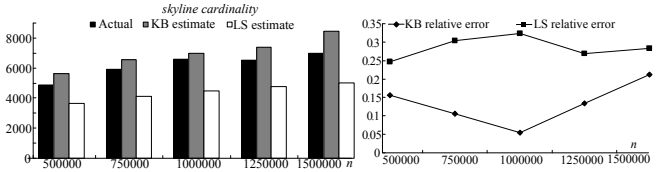
a. Cardinality estimations (*4QP*) b. Error rates (*4QP*)



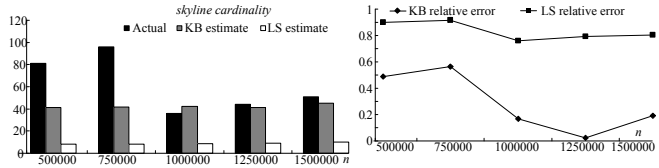
c. Cardinality estimations (*5QP*) d. Error rates (*5QP*)

Figure 16 Accuracy vs. dataset size n (*spatial skyline*)

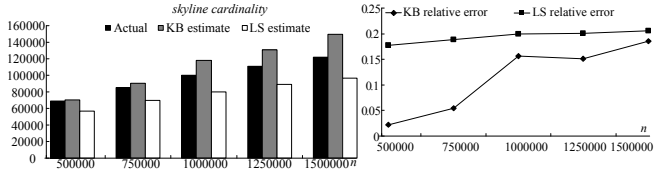
Finally, Figure 17 evaluates synthetic 6D datasets with independent, correlated, and anti-correlated dimensions (n ranges between 0.5-1.5 million). Unlike the previous experiments, for synthetic data with independent and correlated dimensions, ε_{LS} is not affected by n, which agrees with the results of [9]. This suggests that the skyline cardinality for these datasets indeed follows the $A\log^B n$ model. Nevertheless, KB outperforms LS in all settings. Observe that the correlated datasets yield very small skylines. Therefore, although KB and LS have low absolute errors, ε_{KB} and (especially) ε_{LS} are rather high due to the significant impact of random fluctuation. For anti-correlated dimensions, ε_{LS} grows slowly with n, which has also been observed in [9]. This, however, simply suggests that this particular type of anti-correlated data (produced by the generator in [5]) can be captured well by LS because they yield a relatively small skyline (about 10% of the records). On the other hand, recall that for other types of anti-correlated datasets, such as *Corel*, the skyline contains a much higher fraction of the dataset (up to 90% for *Corel*) leading to LS failure.



a. Cardinality estimations (*IND*) b. Error rates (*IND*)



c. Cardinality estimations (*COR*) d. Error rates (*COR*)



e. Cardinality estimations (*ANT*) f. Error rates (*ANT*)

Figure 17 Accuracy vs. dataset size n (*synthetic*)

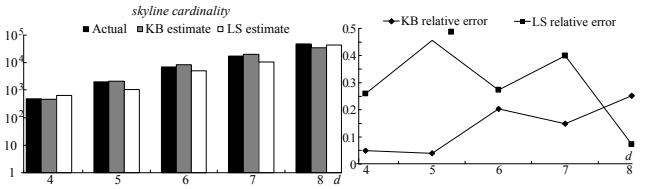
Table 2 lists the median CPU cost of KB, LS in seconds. The two methods incur comparable overhead. The data cardinality does not affect the CPU time because both KB and LS operate on fixed-sized samples. The table does not consider the I/O operations for obtaining the sample sets. Since the sample size of LS is six times larger than that of KB, this omission favors LS.

Table 2 CPU cost of KB, LS and skyline computation

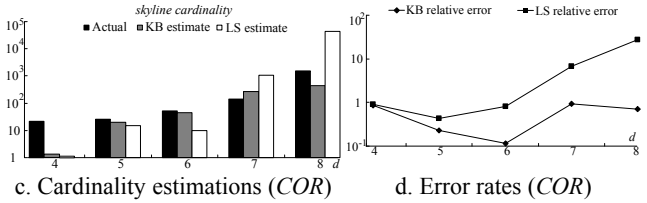
Dataset	KB	LS	Skyline Computation
Household	6.7	5.1	114
Corel	31.4	28.7	607
NBA	12.0	6.3	1327
Spatial 4D	19.5	19.8	> 1 day
Spatial 5D	24.6	20.7	> 1 day
Independent 6D	7.7	4.2	3254
Correlated 6D	1.2	0.9	58
Anti-correlated 6D	32.2	21.5	> 1 day

Skyline estimators, such as KB and LS, are slower than those used for conventional queries (e.g., ranges). However, query processing is accordingly more expensive. In order to illustrate this, we include in the last column of Table 2 the cost of skyline computation using a main-memory nested-loops algorithm [5] that applies several of the optimizations of LESS [15] to eliminate non-skyline points. In all settings, query processing is at least one order of magnitude more expensive than estimation.

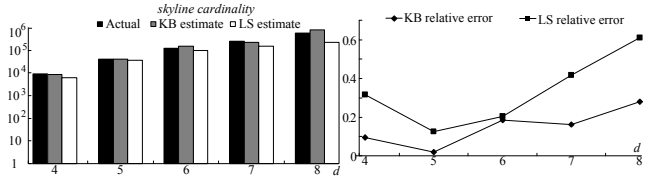
Next, we compare KB and LS with respect to the dimensionality d. Since selecting a subset of attributes in a real dataset may alter its distribution, we focus on synthetic data. Figure 18 demonstrates the impact of d on the accuracy of the two methods. The error rate of KB increases with d, due to the *curse of dimensionality*, which occurs in most kernel-based methods [18]. Nevertheless, KB outperforms LS in all but one (independent 8D) settings. The behavior of LS depends on the dataset characteristics. For independent dimensions, the variation of ε_{LS} is mainly caused by random fluctuations. For correlated and anti-correlated datasets, however, the accuracy of LS deteriorates quickly, indicating that the skyline cardinality shifts away from $A\log^B n$ with increasing d.



a. Cardinality estimations (*IND*) b. Error rates (*IND*)



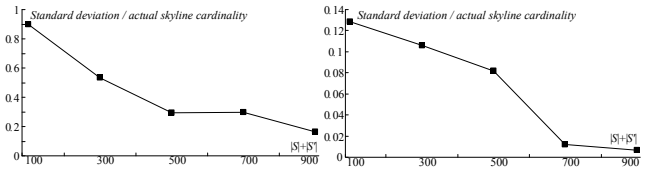
c. Cardinality estimations (*COR*) d. Error rates (*COR*)



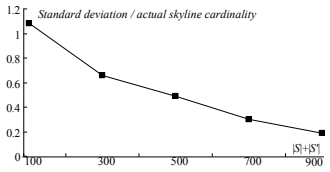
e. Cardinality estimations (*ANT*) f. Error rates (*ANT*)

Figure 18 Accuracy vs. dimensionality *d* (*synthetic*)

Recall that LS extrapolates the skyline cardinality of two sample sets to assess that of the global skyline. Accordingly, the accuracy of LS increases with the sample size as reported in [9]. In contrast, the precision of KB is not directly affected the sample size. In fact, even with very few (in the order of 100) samples, KB sometimes still obtains a highly accurate estimate. On the other hand, the standard deviation of the estimates steadily decreases with an increasing sample size. Figure 19 illustrates the effect of the total sample size $|S|+|S'|$ on the standard deviation of the estimates returned by 10 executions of KB (with different sample sets), in *household*, *Corel* and *NBA*. As expected [6], the standard deviation decreases linearly with $(|S|+|S'|)^{-0.5}$. Results for spatial skylines and synthetic datasets lead to the same conclusions, and are omitted.



a. *Household* b. *Corel*

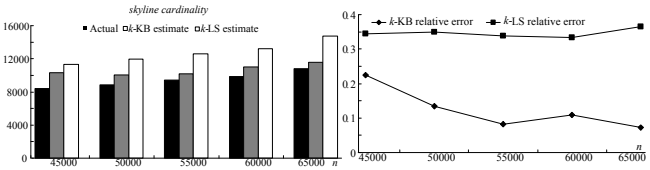


c. *NBA*

Figure 19 Standard deviation of KB vs. sample size

5.2 *k*-Dominant Skyline

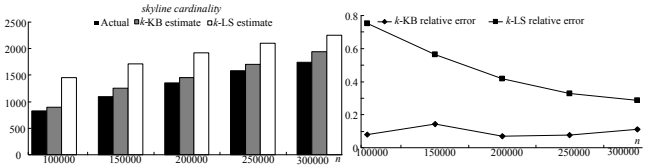
Since k-dominant skylines mainly target high dimensional data, we use *Corel* (9D), *NBA* (16D), and synthetic datasets with 12 dimensions. Due to space limitations, we only vary the data cardinality n and fix k to d–1. Figure 20 evaluates the methods using the *Corel* dataset. Recall from Figure 14 that a large portion of records in *Corel* are in the conventional skyline. The k-dominant skyline, however, has a much higher selectivity, and grows slowly with n, a phenomenon also observed in [8]. Consequently, the error rate ε_{kLS} of k-LS is more stable than ε_{LS}. Furthermore, note that k-LS overestimates the cardinality because it applies an upper-bound of the true value according to Equation (21). k-KB clearly outperforms k-LS in all settings.



a. Cardinality estimations b. Error rates

Figure 20 Accuracy vs. dataset size *n* (*Corel*)

Figure 21 reports the results on NBA. The estimation m_{kLS} of k-LS grows significantly slower than the actual cardinality m with increasing n. Therefore, although ε_{kLS} decreases initially due to reduced overestimation, we expect that as n increases further, m will eventually surpass m_{kLS}, and consequently, ε_{kLS} will start to increase. On the other hand, k-KB consistently achieves high accuracy, and ε_{kKB} is not affected by n.

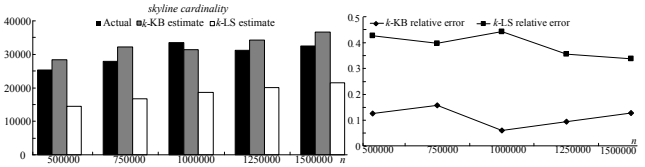


a. Cardinality estimations b. Error rates

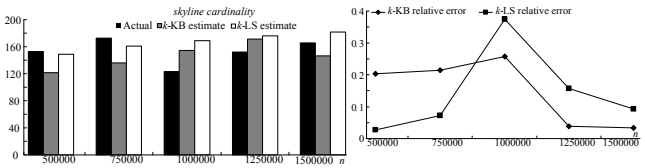
Figure 21 Accuracy vs. dataset size *n* (*NBA*)

Figure 22 investigates synthetic datasets. Both methods achieve relative error rates below 50%. k-KB outperforms k-LS in all but two settings (n = 500K and n = 750K for correlated dimensions). For the independent data, the behavior of k-LS resembles that in *NBA*, in the sense that m_{kLS} grows faster than m with increasing n.

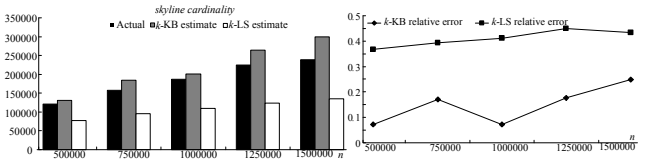
Meanwhile, in the anti-correlated data, m_{kLS} grows slower than m, similar to the situation in *Corel*. For correlated data, both techniques incur low absolute error, and the variance in their relative errors mainly reflects random fluctuations due to the small skyline cardinality.



a. Cardinality estimations (*IND*) b. Error rates (*IND*)



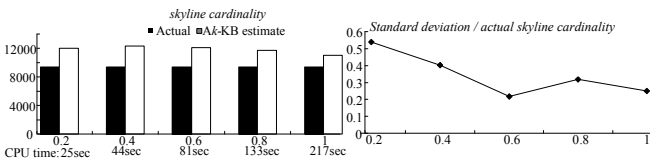
c. Cardinality estimations (*COR*) d. Error rates (*COR*)



e. Cardinality estimations (*ANT*) f. Error rates (*ANT*)

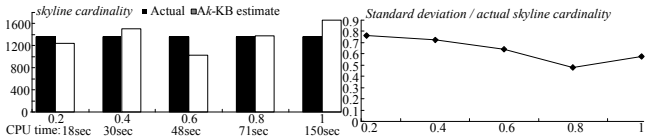
Figure 22 Accuracy vs. dataset size *n* (*synthetic*)

Finally, we evaluate *adjustable k-KB* (A*k*-KB), which provides a tradeoff between efficiency and accuracy through the user-specified parameter τ ($0 \le \tau \le 1$). Figures 23 and 24 demonstrate results for the two real datasets *Corel* and *NBA* as a function of τ, after fixing the dataset sizes to their median values (i.e., 55k in *Corel*, 200k in *NBA*). Specifically, Figures 23a and 24a show the median estimates of A*k*-KB over 20 executions, in comparison with their respective actual values. CPU times are shown under their respective diagrams. Figure 23b and 24b depict the standard deviations as a ratio of the actual skyline cardinality. We observe the following. (i) A low value of τ does not necessarily introduce additional error due to the adaptive nature of A*k*-KB (i.e., the kernel bandwidth is determined iteratively). (ii) The standard deviation generally decreases as τ increases. At the same time, a large number of partitions with small volumes are taken into account. Consequently, the accumulated error in the evaluations of special functions (e.g., *erf*, γ) renders the algorithm less stable, leading to fluctuations in the standard deviation. (iii) The CPU time increases with τ because of the large number of small-volume partitions involved in the computations. Overall, a value of τ between 0.6 and 0.8 achieves a good balance between efficiency and accuracy.



a. Cardinality estimations b. Standard deviations

Figure 23 Effect of τ (*Corel*)



a. Cardinality estimations b. Standard deviations

Figure 24 Effect of τ (*NBA*)

6. CONCLUSION

This paper proposes KB, a kernel-based approach for skyline cardinality estimation. KB is based on nonparametric statistics, and, thus, is more robust compared to LS, the current state of the art technique, which is parametric. Extensive experiments confirm that KB achieves high accuracy in a variety of real and synthetic datasets. As a second step, we extend both LS and KB to the problem of estimating the cardinality of the *k*-dominant skyline, commonly used for high-dimensional data. In the future, we plan to apply KB to other skyline variants, including low-cardinality domains, and continuous monitoring of the skyline cardinality.

ACKNOWLEDGEMENTS

Zhenjie Zhang and Anthony Tung were supported by Singapore ARF grant R-252-000-268-112. Yin Yang and Dimitris Papadias were supported by grant 6184/06 from Hong Kong RGC.

REFERENCES

[1] Bartolini, I., Ciaccia, P., Patella, M. Efficient Sort-based Skyline Evaluation. *TODS*, 33(4):31.1-31.49, 2008.

[2] Bentley, J., Clarkson, K., Levine, D. Fast Linear Expected-Time Algorithms for Computing Maxima and Convex Hulls. *SODA*, 1999.

[3] Bentley, J., Kung, H., Schkolnick, M., Thompson, C. On the Average Number of Maxima in a Set of Vectors and Applications. *Journal of the ACM*, 25(4): 536-543, 1978.

[4] Blohsfeld, B., Korus, D., Seeger, B. A Comparison of Selectivity Estimators for Range Queries on Metric Attributes. *SIGMOD*, 1999.

[5] Börzsönyi, S., Kossmann, D., Stocker, K. The Skyline Operator. *ICDE*, 2001.

[6] Casella, G., Berger, R., *Statistical Inference*. Duxbury Press, 2001.

[7] Chan, C.-Y., Eng, P.-K., Tan, K.-L. Stratified Computation of Skylines with Partially-Ordered Domains. *SIGMOD*, 2005.

[8] Chan, C.-Y., Jagadish, H., Tan, K.-L. Tung, A., Zhang, Z. Finding *k*-dominant Skylines in High Dimensional Space. *SIGMOD*, 2006.

[9] Chaudhuri, S., Dalvi, N., Kaushik, R. Robust Cardinality and Cost Estimation for the Skyline Operator. *ICDE*, 2006.

[10] Chomicki, J. Querying with Intrinsic Preferences. *EDBT*, 2002.

[11] Chomicki, J., Godfrey, P., Gryz, J., Liang, D. Skyline with Presorting. *ICDE*, 2003.

[12] Duda, R., Hart, P., Stork, D. *Pattern Classification*. Wiley-Interscience, 2000.

[13] Fridman, J. Exploratory Projection Pursuit. *Journal of American Statistics Association*, 82:249-266, 1987.

[14] Godfrey, P. Skyline Cardinality for Relational Processing. *FoIKS*, 2004.

[15] Godfrey, P., Shipley, R., Gryz, J. Algorithms and Analyses for Maximal Vector Computation. *VLDB Journal*, 16(1): 5-28, 2007.

[16] Gunopoulos, D., Kollios, G., Tsotras, V., Domeniconi, C. Selectivity Estimators for Multidimensional Range Queries over Real Attributes. *VLDB Journal*, 14(2): 137-154, 2005.

[17] Haas, P., Swami, A. Sequential Sampling Procedures for Query Size Estimation. *SIGMOD*, 1992.

[18] Hwang, J.-N., Lay, S.-R., Lippman, A. Nonparametric Multivariate Density Estimation: A Comparative Study. *IEEE Trans. on Signal Processing*, 42(10): 2795-2810, 1994.

[19] Hwang, J.-N., Lay, S.-R., Lippman, A. Unsupervised Learning for Multivariate Probability Density Estimation: Radial Basis and Projection Pursuit. *IEEE Conf. on Neural Networks*, 1994.

[20] Jagadish, H., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K., Suel, T. Optimal Histograms with Quality Guarantees. *VLDB*, 1998.

[21] Koltun, V., Papadimitriou, C. Approximately Dominating Representatives. *ICDT*, 2005.

[22] Kossmann, D., Ramasak, F., Rost, S. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. *VLDB*, 2002.

[23] Kung, H., Luccio, F., Preparata, F. On Finding the Maxima of a Set of Vectors. *Journal of the ACM*, 22(4): 469-476, 1975.

[24] Lee, K., Zhang, B., Li, H., Lee, W.-C. Approaching the Skyline in Z Order. *VLDB*, 2007.

[25] Lin, X., Yuan, Y., Wang, W., Lu, H. Stabbing the Sky: Efficient Skyline Computation over Sliding Windows. *ICDE*, 2005.

[26] Morse, M., Patel, J., Jagadish, H. Efficient Skyline Computation over Low-Cardinality Domains. *VLDB*, 2007.

[27] Papadias, D., Tao, Y., Greg, F., Seeger, B. Progressive Skyline Computation in Database Systems. *TODS*, 30(1): 41-82, 2005.

[28] Pei, J., Jiang, B., Lin, X., Yuan, Y. Probabilistic Skylines on Uncertain Data. *VLDB*, 2007.

[29] Pei, J., Jin, W., Ester, M., Tao, Y. Catching the Best Views of Skyline: a Semantic Approach Based on Decisive Subspaces. *VLDB*, 2005.

[30] Poosala, V., Ioannidis, Y. Selectivity Estimation without the Attribute Value Independence Assumption. *VLDB*, 1997.

[31] Press, W., Teukolsky, S., Vetterling, W., Flannery, B. *Numerical Recipes in C, Second Edition*. Cambridge University Press, 1992.

[32] Sacharidis, D., Papadopoulos, S., Papadias, D. Topologically-sorted Skylines for Partially-ordered Domains. *ICDE*, 2009.

[33] Sharifzadeh, M., Shahabi, C. The Spatial Skyline Query. *VLDB*, 2006.

[34] Tan, K.-L., Eng, P.-K., Ooi, B.-C. Efficient Progressive Skyline Computation. *VLDB*, 2001.

[35] Tao, Y., Papadias, D. Maintaining Sliding Window Skylines on Data Streams. *IEEE TKDE*, 18(3): 377-391, 2006.

[36] Tao, Y., Xiao, X., Pei J. SUBSKY: Efficient Computation of Skylines in Subspaces. *ICDE*, 2006.

[37] Xia, T., Zhang, D. Refreshing the Sky: the Compressed SkyCube with Efficient Support for Frequent Updates. *SIGMOD*, 2006.

[38] Yuan, Y., Lin, X., Liu, Q, Wang, W., Yu, J. X., Zhang, Q. Efficient Computation of the Skyline Cube. *VLDB*, 2005.