Skyline Cardinality for Relational Processing

How Many Vectors Are Maximal?

Parke Godfrey

York University, Toronto, Ontario M3J 1P3, Canada
godfrey@cs.yorku.ca

Abstract. The *skyline clause*—also called the *Pareto clause*—recently has been proposed as an extension to SQL. It selects the tuples that are Pareto optimal with respect to a set of designated skyline attributes. This is the *maximal vector problem* in a relational context, but it represents a powerful extension to SQL which allows for the natural expression of on-line analytic processing (OLAP) queries and preferences in queries.
Cardinality estimation of skyline sets is the focus in this work. A better understanding of skyline cardinality—and other properties of the skyline—is useful for better design of skyline algorithms, is necessary to extend a query optimizer's cost model to accommodate skyline queries, and helps to understand better how to use skyline effectively for OLAP and preference queries.
Within a basic model with assumptions of *sparseness* of values on attributes' domains and statistical independence across attributes, we establish the expected skyline cardinality for skyline queries. While asymptotic bounds have been previously established, they are not widely known nor applied in skyline work. We show concrete estimates, as would be needed in a cost model, and consider the nature of the distribution of skyline. We next establish the effects on skyline cardinality as the constraints on our basic model are relaxed. Some of the results are quite counter-intuitive, and understanding these is critical to skyline's use in OLAP and preference queries. We consider when attributes' values repeat on their domains, and show the number of skyline is diminished. We consider the effects of having Zipfian distributions on the attributes' domains, and generalize the expectation for other distributions. Last, we consider the ramifications of correlation across the attributes.

1 Introduction

To query relational data to find a best match, the aggregation operators min and max allow one to retrieve the best—that is, either lowest or highest valued—tuples with respect to a single criterion. The order by clause in SQL allows one to rank order the results, perhaps with respect to many criteria.[1] Beyond this, relational query languages as SQL provide little else for finding best matches, or for expressing preferences as part of one's queries.

[1] The rank ordering will be equivalent to a nested sort over the indicated attributes' values.

D. Seipel and J.M. Turull-Torres (Eds.): FoIKS 2004, LNCS 2942, pp. 78–97, 2004.

Consider a table of restaurant guide information, as in Figure 1(a). Column **S** stands for *service*, **F** for *food*, and **D** for *decor*. Each is scored from 1 to 30, with 30 as the best.[2] We are interested in choosing a restaurant from the guide, and are looking for a best choice, or a set of best choices from which to choose. Ideally, we would like the restaurant chosen to be the best for service, food, *and* decor, *and* be the lowest priced. There is likely no restaurant that is better than *all* others on *all* criteria, however.[3] No one restaurant trumps all others. For instance, Summer Moon is best on food, but Zakopane is best on service.

restaurant	**S**	**F**	**D**	**price**
Summer Moon	21	25	19	47.50
Zakopane	24	20	21	56.00
Brearton Grill	15	18	20	62.00
Yamanote	22	22	17	51.50
Fenton & Pickle	16	14	10	17.50
Briar Patch BBQ	14	13	3	22.50

(a) Restaurant guide table, GoodEats.

restaurant	**S**	**F**	**D**	**price**
Summer Moon	21	25	19	47.50
Zakopane	24	20	21	56.00
Yamanote	22	22	17	51.50
Fenton & Pickle	16	14	10	17.50

(b) The skyline.

Fig. 1. The restaurant table and the skyline.

While there is no one best restaurant, we can eliminate from consideration any restaurant that is worse on all the criteria than another. The Briar Patch BBQ should be eliminated because the Fenton & Pickle is better in comparison across all our criteria. The Brearton Grill is eliminated, in turn, because Zakopane is better than it on all criteria. If Zakopane were not in the table, the Brearton Grill would have remained a consideration. Meanwhile the Fenton & Pickle is worse on every criterion than every other (remaining) restaurant, except on price, where it is the best. So it stays in consideration. (If we were to remove price as one of our criteria, the Fenton & Pickle would be eliminated too.) This would result in the choices in Figure 1(b).

In [2], an extension to SQL is proposed, the skyline of clause, which allows easy expression of the restaurant query we imagined above. They propose also the *skyline operator* as the relational algebraic counterpart of the clause, and pursue an implementation to support efficiently skyline queries within a relational environment.

The skyline of clause is shown in Figure 2(a). It is syntactically similar to an order by clause. Columns $a_1, \ldots, a_n$ are the attributes over which our preferences apply. Their domains must have a natural total ordering, such as integers, floats, and dates. The directives min and max specify whether we prefer low or high values, respectively. The directive diff states that one wants to retain best choices

[2] This table is modeled on the Zagat Survey Guides. For example, see [1].

[3] This is certainly the case in real life, and in real data!

```
select ... from ... where ...
    group by ... having ...
    skyline of a1 [min | max | diff], ...,
               an [min | max | diff]
```

(a) Skyline clause for SQL.

```
select * from GoodEats
    skyline of S max, F max,
               D max, price min
```

(b) Choosing restaurants.

Fig. 2. Skyline queries.

with respect to each distinct value of that attribute. Let min be the default directive, if none is stated. The skyline query in Figure 2(b) over the table GoodEats in Figure 1(a) expresses what we had in mind above for choosing "best" restaurants, and would result in the answer set in Figure 1(b). If the table GoodEats had a column **C** for *cuisine*, we could add C diff to the skyline of clause to find the best restaurants by each cuisine group.

```
select c1, ..., ck, s1, ..., sm, d1, ..., dn
    from OurTable
except
select D.c1, ..., D.ck, D.s1, ..., D.sm,
       D.d1, ..., D.dn
    from OurTable T, OurTable D
    where D.s1≥ T.s1 and ... D.sm≥ T.sm and
        (D.s1> T.s1 or ... D.sm> T.sm) and
        D.d1= T.d1 and ... D.dn= T.dn
```

Fig. 3. SQL for generating the skyline set.

Skyline queries are not outside the expressive power of present SQL. The query in Figure 3 demonstrates one way to write an arbitrary skyline query in present SQL. The c_i's are attributes of OurTable that we are interested to retain in our query, but that are not skyline criteria. The s_i are the skyline attributes to be minimized, and would appear in skyline of as s_i min. (Without loss of generality, we only consider min and not max.) The d_i are the attributes that are the skyline criteria to *differ*, and would appear in skyline of as s_i diff. It is cumbersome to write skyline-like queries currently in SQL, however. The skyline clause would be a useful syntactic addition to SQL, therefore, if skyline-like queries were to become commonplace. More important than ease of expression, however, is the expense of evaluation. The query in Figure 3 can be prohibitively expensive. It involves a self-join over a table. This join is a θ-join, not an equality-join. It effectively computes the tuples that are trumped—or *dominated*—by other tuples. The tuples then that remain—that were never dominated—determined by the except operation, constitute the skyline tuples.

There has been a fair amount of recent work focused on how to compute efficiently skyline queries within relational systems for large datasets. In Section 2, we discuss the related work. Skyline cardinality estimation is the focus of this work. A better understanding of skyline cardinality

- is useful for better design of skyline algorithms,
- is necessary to extend the query optimizer's cost model to accommodate skyline queries, and
- helps us to understand better how to use skyline effectively for preference queries.

In Section 3, we present a *basic model*—with assumptions of sparseness over attributes' domains (namely that there are virtually no duplicate values) and statistical independence across attributes—and devise concrete estimates of skyline query cardinalities under the basic model. We consider next the distribution of the number of skyline. The expected value would not be of much utility if the distribution were not to be well behaved. We show through Monte Carlo simulations the nature of the distribution, and that it is well behaved.

In Section 4, we consider the effects on the estimates and distributions as we relax the assumptions of the basic model, thus modeling better the characteristics of real data. We consider the effects of when attribute domains are restricted to small domains (allowing repeated values and ties), different distributions over attribute domains (namely, Zipfian), and pair-wise correlation between attributes. Under the basic model, the number of skyline tuples is *independent* of the distributions of the attributes' domains. This is no longer necessarily true when attributes' values are no longer sparse over their domains. When there are duplicate values over the attributes, the number of skyline tuples *decreases*. We show that the skyline estimates of the basic model are a *ceiling* for skyline cardinalities when the sparseness condition is violated.[4]

In Section 5, we discuss briefly some of the ramifications of our results for skyline algorithm design, cost models for the skyline operator, and the use of skyline in preference queries, and we conclude. We believe these studies can also offer general insights into queries over multi-dimensional criteria.

2 Related Work

The concept of skyline in itself is not new, of course. The search for optimal solutions is a well-established endeavor with an exceedingly deep literature. Beginning in the 1960's, work focused on optimization with respect to multiple criteria. Techniques have been explored for finding good *utility functions* to combine effectively the multiple criteria of interest into a single score. Then traditional mathematical techniques for finding the optimal solution—with respect to a single criterion, the utility function in this case—could then be applied. Others recognized though that for many applications it is often difficult—if not virtually impossible—to find a reasonable utility function. Thus work in *multiple*

[4] There is an exception case, to be discussed.

criteria optimization focused on how to find *all* optimal solutions in the space with respect to the multiple criteria [3]. The definition of solution in this context is often identical to our definition for skyline: no other potential solution is better across all the criteria. This is called *Pareto optimal*.

The problem attracted attention early on within the mathematics and statistics communities, and within the computational geometry community, since the Pareto optimal points are a super-set of the points that delineate the convex hull of the entire set. How to compute the convex hull, and its size, is of central interest in linear optimization problems. Perhaps in [4] is the first work to study the distribution of *admissible* (Pareto optimal) points, and the expected value of their number. More recently in [5], the work of [4] is extended to show bounds on the variance of the number of admissible points. In [6], Golin considers the effects of the shape of the space to which the points are restricted, and how that shape changes the expected value of the number of admissible points.

Multiple-criterion optimization usually assumes an implicit solution space from which the optimal solutions are to be found. Often, this space is quite large, but also has properties that help to devise good techniques. For skyline queries, the space is explicit: it is the input relation of vectors, or tuples. One does not know necessarily particular properties of the space.

The skyline idea has been studied before in this context of an explicit solution space as the *maximal vector problem*. In [7], the first algorithm to find the maximal vectors (or skyline tuples) from a set of vectors (or relation) was devised. In [8], the maximal vector problem is addressed in the context of computational geometry. In [9], they established that the average number of skyline tuples is $\mathcal{O}((\ln n)^{d-1})$.[5] This is the cardinality bound most often cited and employed in skyline work. However, this is a loose upper-bound. In [10], it is established that $\Theta((\ln n)^{d-1}/(d-1)!)$. There has been work also to establish theoretical complexity bounds for computing the maxima (the Pareto points) and those points on the convex hull [11,12]. This work does not lead directly to good algorithms in practice, however, for computing skyline queries.

Interest has returned to the maximal vector problem recently indeed in the guise of skyline queries. Previous work was main-memory based though, and not well suited to databases. Progress has been made recently on how to compute efficiently such queries in a relational system and over large datasets [2,13,14, 15,16]. In [2], the skyline operator is introduced. They posed two algorithms for it, a block-nested loops style algorithm (and variations) and a divide-and-conquer approach derived from work in [7,8]. In [14,15,16], algorithms for skyline evaluation that exploit indexes are developed. In [13], we developed a general skyline algorithm, based on the "block-nested loops" algorithm of [2], which is more efficient, pipelinable, and amenable to use in a relational query optimizer.

A related topic is nearest-neighbor search. This has been studied in the context of relational systems too [17]. In [18], elements of a cost model for nearest-neighbor searches are considered, but just for high-dimensional cases. In [19],

[5] For this, they made essentially the same assumptions that we shall make in Definition 3.1 about attributes' distributions and independence.

a specialized index structure is developed to facilitate nearest-neighbor queries. In [14], they employ nearest-neighbors algorithms to pipeline the generation of skyline tuples.

Interest in skyline queries arises in most part from the desire to support queries with preferences in relational systems. In [20], a more general operator called *winnow* is introduced for the purpose of expressing preference queries. Skyline is a special case of winnow. Skyline and related techniques could make it possible to integrate easily certain *cooperative query answering*, *query relaxation*, and *preference query* techniques which have been proposed [21,22,23,24]. In [25], a framework is presented for how to express and combine effectively preferences in queries. In [26], a system called PREFER is presented which is designed to handle multi-parametric ranked queries efficiently. In [27], a preference algebra is developed along with extensions to SQL for general preference queries. They call for a more efficient means to compute preference queries. In [28], a system that incorporates the preference SQL of [27] is presented.

3 Pareto / Skyline Cardinality

We want to estimate the cardinality of the output relation of the skyline operator based upon its input relation. The input can be a base table of the database or a virtual table which is the intermediate result in a query's evaluation. Let us establish a basic model of assumptions about the input relation under which it is possible to establish analytically the cardinality.

Definition 3.1. Basic model of the input relation and skyline query.

Let dimension *refer to an attribute of the relation that participates in the skyline criteria.*

a. Domain assumption (sparseness): *For each dimension, we assume that there are no duplicate values on the attribute across the tuples of the relation.*

b. Independence assumption: *The dimensions are statistically independent.*[6]

Consider a skyline operation with d dimensions over such an input relation of n tuples. (So the relation has at least d attributes, which obey the assumptions above.) Let $\mathsf{s}_{d,n}$ *be the random variable which measures the number of tuples (the cardinality) of the output relation (that is, the set of the resulting skyline tuples). Let* $\hat{\mathsf{s}}_{d,n}$ *denote the expected value of* $\mathsf{s}_{d,n}$*.*

We are interested to know $\hat{\mathsf{s}}_{d,n}$. Under our basic model, no two input tuples share a value over any dimension. Thus, the tuples can be ordered totally on any given dimension. It is not necessary therefore to consider the actual values of the tuples. Instead, we can conceptually replace the value on, say, dimension i of a tuple by its *rank* in the total ordering along dimension i. Without loss of generality, let us assume that we are *minimizing* over the dimensions for the skyline. Let rank 1 refer to the tuple with the smallest value (on that dimension), and n the one with the largest (n is the number of input tuples). We can now just

[6] That is, there are no pair-wise or group correlations nor anti-correlations.

refer to a tuple's rank on a dimension and ignore the actual value. We provide the proof as it is illustrative for the following discussion.

Theorem 3.1. *[9] The skyline expected value* $\hat{\mathsf{s}}_{d,n}$ *for* $d > 1$ *and* $n > 0$ *obeys the following recurrence equation.*

$$\hat{\mathsf{s}}_{d,n} = \frac{1}{n}\hat{\mathsf{s}}_{d-1,n} + \hat{\mathsf{s}}_{d,n-1}$$

For $n > 0$, $\hat{\mathsf{s}}_{1,n} = 1$.

Proof. *Consider* $\hat{\mathsf{s}}_{1,n}$. *Since no two tuples share the same value on the dimension, only the tuple with rank* 1 *is in the skyline.*

Consider $\hat{\mathsf{s}}_{d,n}$, *for* $d > 1$. *One tuple has rank* n *on dimension* 1. *This tuple cannot dominate any other tuple, since it has a higher value on dimension* 1 *than any other. What is the probability that this tuple itself is a skyline tuple? It is the probability that no other tuple dominates it on dimensions* $2, \ldots, d$, *given the independence assumption. As* $\hat{\mathsf{s}}_{d-1,n}$ *is the expected value of the number of skyline tuples out of* n *tuples on* $d-1$ *dimensions, then* $\frac{1}{n}\hat{\mathsf{s}}_{d-1,n}$ *represents the probability that this one tuple is part of the skyline.*

Since the n-th *ranked tuple on dimension* 1 *cannot dominate any other tuple, the estimated number of skyline tuples of the remaining* $n-1$ *is* $\hat{\mathsf{s}}_{d,n-1}$. □

The recurrence for $\hat{\mathsf{s}}_{d,n}$ is related to the harmonic numbers.

Definition 3.2. Harmonic numbers.

a. The harmonic of n, *for* $n > 0$: $\mathsf{H}_n = \sum_{i=1}^{n} \frac{1}{i}$

b. [29] The k-th order harmonic of n, *for integers* $k > 0$ *and integers* $n > 0$: $\mathsf{H}_{k,n} = \sum_{i=1}^{n} \frac{\mathsf{H}_{k-1,i}}{i}$

Define $\mathsf{H}_{0,n} = 1$, *for* $n > 0$. *Define* $\mathsf{H}_{k,0} = 0$, *for* $k > 0$.

c. The k-th hyper-harmonic of n, *for integers* $k > 0$ *and integers* $n > 0$: $\mathcal{H}_{k,n} = \sum_{i=1}^{n} \frac{1}{i^k}$

A common two-parameter generalization of the harmonic series is that of the hyper-harmonics in Definition 3.2(c). The $\mathcal{H}_{k,n}$ converge for $k > 1$. A second two-parameter generalization is given in Definition 3.2(b), introduced in [29]. (What is effectively a generalization of the $\mathsf{H}_{k,n}$'s appears in [30], in Section 1.2.9.) The $\mathsf{H}_{k,n}$ do not converge for $k > 1$. Rather, $1 \leq \mathsf{H}_{k,n} < n$, for $k > 0$ and $n > 0$. Furthermore,

$$\hat{\mathsf{s}}_{d+1,n} = \mathsf{H}_{d,n} = \sum_{i_1=1}^{n} \sum_{i_2=1}^{i_1} \cdots \sum_{i_d=1}^{i_{d-1}} \frac{1}{i_1 i_2 \cdots i_d}$$

Any particular $\mathsf{H}_{k,n}$ can be solved in terms of $\mathcal{H}_{j,n}$'s ($1 \leq j \leq k$). The H_n and $\mathcal{H}_{k,n}$ are easy to compute, or approximate, and so could be used within a cost model on-the-fly. (Note that $\mathsf{H}_n = \mathcal{H}_{1,n}$.) For instance, we can derive that

- $\mathsf{H}_{2,n} = \frac{1}{2}\mathsf{H}_n^2 + \frac{1}{2}\mathcal{H}_{2,n}$,
- $\mathsf{H}_{3,n} = \frac{1}{6}\mathsf{H}_n^3 + \frac{1}{2}\mathsf{H}_n\mathcal{H}_{2,n} + \frac{1}{3}\mathcal{H}_{3,n}$, and
- $\mathsf{H}_{4,n} = \frac{1}{24}\mathsf{H}_n^4 + \frac{1}{3}\mathsf{H}_n\mathcal{H}_{3,n} + \frac{1}{8}\mathcal{H}_{2,n}^2 + \frac{1}{4}\mathsf{H}_n^2\mathcal{H}_{2,n} + \frac{1}{4}\mathcal{H}_{4,n}$.

This can be generalized as follows.

Theorem 3.2. $\mathsf{H}_{k,n} = \sum_{\substack{c_1,\ldots,c_k \geq 0 \ \wedge \\ 1\cdot c_1 + 2\cdot c_2 + \ldots + k\cdot c_k = k}} \prod_{i=1}^{k} \frac{\mathcal{H}_{i,n}^{c_i}}{i^{c_i} \cdot c_i!}$ *for* $k \geq 1$ *and* $n \geq 1$, *with the* c_i*'s as integers.*

Proof. *This follows from Knuth's generalization via generating functions [30].* □

For any k, the coefficients of the terms in the summation sum to one. The number of terms to express $\mathsf{H}_{k,n}$ in terms of $\mathcal{H}_{j,n}$'s is $\wp(k)$, the number of ways to partition the positive integer k as a sum of positive integers. Since $\wp(k)$ grows quickly—for instance, $\wp(10) = 42$ and $\wp(20) = 627$—it is not a viable to solve for $\mathsf{H}_{k,n}$'s in this way.

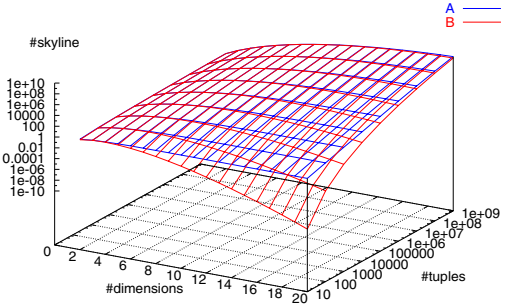


Fig. 4. Plot of $\hat{\mathsf{s}}_{d,n}$ (A) and Buchta's (B).

From this though, we can prove that $\mathsf{H}_{k,n}$ is $\Theta(\mathsf{H}_n^k/k!)$, or equivalently, $\Theta((\ln n)^k/k!)$. In [10], Buchta established that

$$\hat{\mathsf{s}}_{k+1,n} = \frac{(\ln n)^k}{k!} + \frac{\gamma \cdot (\ln n)^{k-1}}{(k-1)!} + \mathcal{O}(\ln n)^{k-2}$$

and therefore that $\hat{\mathsf{s}}_{d,n}$ is $\Theta((\ln n)^{d-1}/(d-1)!)$. We can use these as approximations for $\hat{\mathsf{s}}_{d,n}$. In Figure 4, we plot Buchta's "approximation" (setting the multiplicative constant at one and the additive constant at zero for the $\mathcal{O}$-term) and the actual $\mathsf{H}_{k,n}$ values. $\mathsf{H}_n^k/k!$ and Buchta's underestimate $\mathsf{H}_{k,n}$, with Buchta's

being marginally closer. Interestingly, while $\mathsf{H}_n^k/k!$ and Buchta's are both monotonically increasing with respect to n, neither is with respect to k. So neither is a good concrete estimate for higher dimensions where the divergence becomes more significant.

Of course, the $\mathsf{H}_{k,n}$ can be numerically approximated off-line, and then a look-up table used at run-time. We computed $\mathsf{H}_{d-1,n}$ for $d = 2, \ldots, 20$ and $n = 10, \ldots, 10^7$ by factors of ten, as appears in Figure 4. As with logarithms, interpolation can be safely used to estimate values that are not in the look-up table.

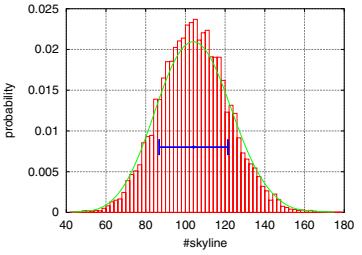
We have the expected value of skyline cardinality, $\hat{\mathsf{s}}_{d,n}$ (with respect to basic model in Definition 3.1), but we do not know the distribution of the random variable $\mathsf{s}_{d,n}$. If its variance were huge, for instance, our use of $\hat{\mathsf{s}}_{d,n}$ in a cost model for a query optimizer would be of limited utility. It would also be possible for the median to be less than the mean, with a long tail towards n. We are really interested in likely values the optimizer will encounter, not the *expected value*, per se, which itself as a value might never occur. In a cost model, it is important to anticipate the chance of being significantly off the estimation, especially in the case when the actual cardinality is exceedingly larger than the estimate. The query plan can be made to accommodate contingencies, to varying degrees.

For the case of $d = 2$, it can be proven that the distribution tends to Gaussian by the Central Limit Theorem. (A proof is sketched in [4].) For $d > 2$, the nature of the distribution remains unknown [4]. There is speculation that the distributions for $d > 2$ also tend to Gaussian, and we present experimental support for this, but it remains unproven.

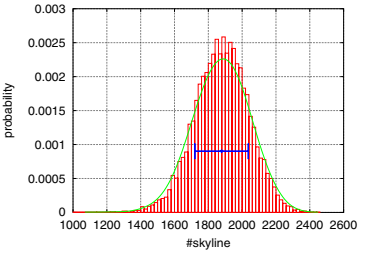
It is possible to infer some properties of the distribution just given what we know of $\hat{\mathsf{s}}_{d,n}$. The domain of $\mathsf{s}_{d,n}$ is $1 \ldots n$, of course. For d and n combinations that are likely for skyline queries in practice, $\hat{\mathsf{s}}_{d,n}$ is close to the 1-end of the spectrum. (See Figure 5(d).) This statistically limits how large the variance can be. (There cannot be much probability that $\mathsf{s}_{d,n}$ is near n, since this would serve to inflate $\hat{\mathsf{s}}_{d,n}$, as by Chebychev's Inequality.)

We study experimentally the distribution. We ran Monte Carlo simulations as follows. For each *trial*, we generated one million tuples of d attributes randomly. Each attribute was of type integer and its value was randomly chosen across all values.[7] The number of skyline tuples (minimizing over the d values) was then determined. A simulation then consisted of $10,000$ trials. Figures 5(a), (b), and (c) show the distributions of the $10,000$ trials for $\mathsf{s}_{d,10^6}$ for $d = 3$, 5, and 7, respectively. The data points were binned into about sixty bins in each case to approximate the distribution via a histogram. We normalize the y-axis to probability so the area covered by the histogram is one. In each case, the error-bar represents the mean and spans one standard deviation to the left and to the right of the mean. The super-imposed curve is a bezier fit of the data. The distributions of $\mathsf{s}_{d,n}$ are well behaved and resemble normal (Gaussian) distributions. That the medians are so near the means in the simulations offers evidence that the true distributions have little if no skew.

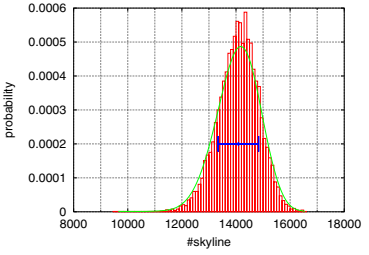
[7] The values range over 1, ..., 2,147,483,647.



(a) Distribution of $\mathsf{s}_{3,10^6}$.



(b) Distribution of $\mathsf{s}_{5,10^6}$.



(c) Distribution of $\mathsf{s}_{7,10^6}$.

	$\mathsf{H}_{d-1,10^6}$	Monte Carlo simulations		
d	$\hat{\mathsf{s}}_{d,10^6}$	μ	median	σ
3	104	104	104	17.3
5	1,880	1,878	1,882	158.5
7	14,087	14,087	14,120	745.9

(d) Means, medians, and standard deviations.

Fig. 5. Distributions of $\mathsf{s}_{d,10^6}$.

In [4], they too are interested in the variance of the number of maxima. They establish that the variance of $\mathsf{s}_{d,n}$ converges from above on the expected value, $\hat{\mathsf{s}}_{d,n}$, for any fixed d as n grows. However, this convergence must be extremely slow, as we see. Still, this is further proof that the variance is quite well behaved, and only becomes better behaved as n becomes larger.

4 Generalizing the Basic Model

We explore next the effects of relaxing the assumptions of our basic model from Definition 3.1.

First, we consider the ramifications of relaxing the sparseness condition. Thus tuples may now match on values, and in the extreme, two tuples may be equivalent with respect to their skyline attributes. A tuple is skyline (under min criteria) if there is no other tuple that has a smaller *or equal* value on each skyline attribute *and* has, for some skyline attribute, a strictly smaller value. Therefore, there may be duplicates among the skyline. Such data "density" of course is characteristic of much real data. Denseness does affect skyline, but in a way

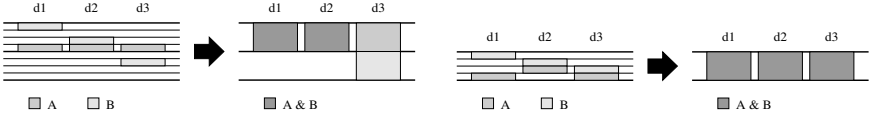
that is counter-intuitive to most. The primary effect is to *reduce* the number of skyline with respect to $\hat{\mathsf{s}}_{d,n}$, the estimated value under the sparseness condition.

So far, we have been unconcerned about the distributions of the tuples' values over each skyline column. We observe that, under sparseness, the distributions are immaterial![8] This is a remarkable characteristic of the skyline. However, under denseness, the distributions do have an effect. We consider Zipfian distributions on the columns and explain what happens. Again, the primary effect is either to reduce the number of skyline with respect to $\hat{\mathsf{s}}_{d,n}$, or not to affect it virtually at all.

Last, we consider the effects of correlation and anti-correlation among the columns. We know *a priori* that correlation must be well behaved with respect to skyline cardinality, but that anti-correlation can move the skyline count to n (with *all* the tuples being in the skyline). Yet we find that skyline cardinality is fairly stable with $\hat{\mathsf{s}} \ll n$, even in the presence of reasonably high anti-correlation.

4.1 Sparseness versus Denseness

Many attribute domains are small. For instance, a *Boolean* type only allows the values *true* and *false*. Furthermore, we are really interested in the *range* of values that occur over an attribute, rather than the domain, per se. For a given distribution of tuples, the values may cluster on just a few of the possible values. When we relax the sparseness assumption from Definition 3.1, it allows for tuples to share values. Furthermore, it allows for duplicate tuples (at least with respect to the skyline attributes). Since most real data is like this, we are interested in how this affects the skyline cardinality.



(a) Two tuples become comparable. (b) Two tuples become identical.

Fig. 6. Binning effects.

Definition 4.1. *Let* $\mathsf{s}_{\langle p \times p \rangle, n}$ *denote the random variable that measures the number of skyline tuples from a relation of* n *tuples, with respect to a* two-*dimensional skyline query. Each of the skyline attributes range over* p *values, the tuples' values are uniformally distributed over them, and the skyline attributes are statistically independent.*

[8] This is with the assumption still of statistical independence over the attributes. Note that the attributes' distributions *can* cause statistical dependence even if the attributes remain causally independent.

Let $\mathsf{s}_{\langle p^d \rangle, n}$ *more generally denote the number of skyline tuples with respect to a d-dimensional skyline query under the same conditions. Let* $\hat{\mathsf{s}}_{\langle p^d \rangle, n}$ *denote the expected value.*

It is straightforward to establish $\hat{\mathsf{s}}_{\langle p^d \rangle, n}$'s behavior in the limit, for d, p, and n.

Theorem 4.1. Bounds on $\mathsf{s}_{\langle p^d \rangle, n}$ and $\hat{\mathsf{s}}_{\langle p^d \rangle, n}$.

a. $1 \leq \mathsf{s}_{\langle p^d \rangle, n} \leq n$

b. $\lim_{d \to \infty} \hat{\mathsf{s}}_{\langle p^d \rangle, n} = n$

c. $\lim_{p \to \infty} \hat{\mathsf{s}}_{\langle p^d \rangle, n} = \hat{\mathsf{s}}_{d,n}$

d. $\lim_{n \to \infty} \dfrac{\hat{\mathsf{s}}_{\langle p^d \rangle, n}}{n} = \dfrac{1}{p^d}$

Proof. *There must be at least one skyline tuple, and at most, there can be n. In the limit of d, all tuples will be incomparable. In the limit of p, the probability of repeat values diminishes to zero. Once* $n \gg 1/p^d$*, with high probability, there will be tuples with the highest value on each dimension. Just these tuples, all duplicates with respect to skyline attributes, will be skyline.* □

We are specifically interested in how $\hat{\mathsf{s}}_{\langle p^d \rangle, n}$ relates to $\hat{\mathsf{s}}_{d,n}$. Does value repetition (denseness) increase the number of expected skyline tuples, or decrease it? Denseness is equivalent to considering a relation that is initially sparse, and the tuples' values are then *binned*—that is, *partitioned*—over each attribute into just a few bins (values). So the tuples initially share no values, but after being *binned*, they do. Of course after binning, there can be duplicate *tuples* as well. Figure 6 shows two effects that occur. In some cases, a pair of tuples that were incomparable before binning may be comparable after binning. Figure 6(a) shows this.[9] Tuples A and B were incomparable before. Assume that they both are skyline. However, after binning, A trumps B. Thus only A could be in the skyline of the binned relation.[10] By this effect, it is possible to lose skyline tuples. In other cases, *all* the skyline attribute values of the two tuples become the same. Figure 6(b) shows this. Assume that A is skyline but B is not initially. Again, two incomparable tuples have become comparable upon binning, but this time both A and B might qualify as skyline afterward. By this second effect, it is possible to gain skyline tuples, due to resulting duplicate tuples.

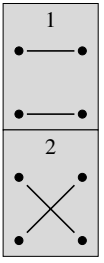
The probability of value sharing occurring, as in Figure 6(a), is much greater generally than that of the tuples becoming equivalent, as in Figure 6(b).[11] That is, there is a higher probability that tuples will share values on some dimensions, but not on all. This gap increases with the number of dimensions. So we generally expect the first effect to dominate the second, and for the number of skyline to go down due to binning.

[9] The vertical axis represents the attributes' values. The horizontal axis spans the attributes. The two tuples, A and B, are shown in different shades.

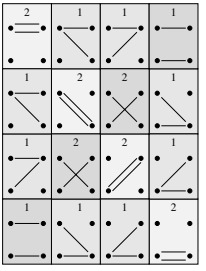
[10] We say "could" because some other tuple might trump A after binning by this same argument.

[11] This is not the case only in the extreme. For instance, if we bin all values to a single bin for each dimension, all the tuples become identical.

There is the case when there are many more tuples, n, than possible value combinations, p^d. In this case, the second effect due to duplicates reigns. By uniformity, for each possible value combination, there is with high probability a tuple that matches it. Thus, there is a tuple with the best p value on each attribute, and so this is the only possibility for skyline. How many skyline tuples there are depends on how many duplicates of this best tuple there are. The limit in Theorem 4.1(d) reflects this effect.



(a) Edges with no repeats, $\hat{\mathsf{s}}_{2,2} = 3/2$.



(b) Edges with repeats, $\hat{\mathsf{s}}_{2\times 2,2} = 11/8$.

Fig. 7. Choose two edges.

Consider the case of two tuples of two dimensions, with each dimension as Boolean. This is the same as considering two edges placed on a 2×2 bipartite grid, as in Figure 7. If no vertex (value) sharing is allowed, the only possibilities are as in Figure 7(a). With vertex sharing, there are sixteen possibilities as in Figure 7(b) (choosing two possible edges, with replacement). By our assumptions of uniform distributions and independence, all sixteen possibilities are equally likely.

The dark-hued diagonal in Figure 7(b) represents the cases in which there are no repeated values. These behave exactly as $\mathsf{s}_{2,2}$. The light-hued diagonal are the duplicate cases, so $\hat{\mathsf{s}} = 2$ over these. The rest are cases of repeats, but no duplicates. For these, $\hat{\mathsf{s}} = 1$. Note there are twice as many cases of repeats (but no duplicates) than of duplicates.

We can solve via the probabilities for $\hat{\mathsf{s}}_{\langle p\times p\rangle,2}$ (and for higher values of n, although it becomes increasingly cumbersome).

Lemma 4.1. $\hat{\mathsf{s}}_{\langle p\times p\rangle,2} = \mathsf{H}_2 - \dfrac{1}{p} + \dfrac{3}{2p^2}$

Proof. *Follows from a case-by-case sum of the probabilities.* □

For $p > 1$, $\hat{\mathsf{s}}_{\langle p\times p\rangle,2} < \hat{\mathsf{s}}_{2,2}$.

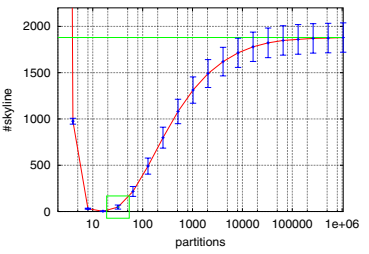
We can establish more generally that

Lemma 4.2. $\hat{\mathsf{s}}_{\langle p^d\rangle,2} < \hat{\mathsf{s}}_{d,2}$, *for* $p \geq 2$ *and* $d > 1$.

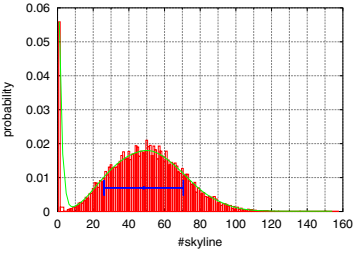
Proof. *We show* $\hat{\mathsf{s}}_{\langle p^d\rangle,2} \leq 2-1/2^{d-2}+(1/p^d)(1/2^{d-2})$ *and that* $\hat{\mathsf{s}}_{d,2} = 2-1/2^{d-1}$. *Thus,* $\hat{\mathsf{s}}_{\langle p^d\rangle,2} < \hat{\mathsf{s}}_{d,2}$ *when* $p \geq 2$ *and* $d > 1$. □

We conjecture that this is true more generally for any n, up to a point before n saturates the number of admissible values.

Conjecture 4.1. For $n < p^d \cdot \hat{\mathsf{s}}_{d,n}$, $\hat{\mathsf{s}}_{\langle p^d\rangle,n} < \hat{\mathsf{s}}_{d,n}$.



(a) Plotting "$\hat{\mathsf{s}}_{\langle p^5\rangle,10^6}$" for $d = 5$.



(b) Distribution of "$\mathsf{s}_{\langle 32^5\rangle,10^6}$".

Fig. 8. $\hat{\mathsf{s}}_{\langle p^5\rangle,10^6}$

This conjecture may be hard to prove, especially for cases of small n's and p's. For instance, while $\hat{\mathsf{s}}_{\langle p\times p\rangle,2} < \hat{\mathsf{s}}_{2,2}$ (for $p > 1$), $\hat{\mathsf{s}}_{\langle p\times p\rangle,2}$ is not monotone with respect to p. Of course, for our purposes, we are only interested in relatively large n. To ascertain experimentally $\hat{\mathsf{s}}_{\langle p^d\rangle,n}$ and the distribution of $\mathsf{s}_{\langle p^d\rangle,n}$, as in Section 3, we ran 10,000 trials for each of $p = 2$ to 2^{20} (by doubling) for $d = 5$ and $n = 10^6$. Figure 8(a) shows this. The error-bars represent the standard deviations. Figure 8(b) shows the distribution of $\mathsf{s}_{\langle 32^5\rangle,10^6}$, and is constructed in the same way as those in Figures 5(a), (b), and (c).

To the left of the nadir in Figure 8(a), the number of tuples dominates the possible value combinations, and the distribution counts the number of duplicates of that best scoring combination. These distributions are true Gaussian, by the Central Limit Theorem. (For these, we note that $\mu = \sigma^2$.) Around the nadir is the balance point between the duplicate effect and value sharing. Here, the distributions are odder, as in Figure 8(b). That distribution is bimodal, in which many trials hit the "best" value combination once (and so have a skyline of one) and many do not. As we move to the right, the distributions become well behaved and Gaussian-like again. The number of skyline is diminished due to the value-sharing effect, which acts as dimensional reduction. The distributions converge on that of $\mathsf{s}_{5,10^6}$ (1,880)—shown as the ceiling in Figure 8(a)—as p grows and the relation becomes virtually sparse.

More generally, the distribution of $\mathsf{s}_{\langle p^d,n\rangle}$ converges on that of $\mathsf{s}_{d,n}$ in the limit as p grows, and $\hat{\mathsf{s}}_{\langle p^d,n\rangle}$ converges asymptotically from below to $\hat{\mathsf{s}}_{d,n}$. Thus, $\hat{\mathsf{s}}_{d,n}$ is a ceiling on $\hat{\mathsf{s}}_{\langle p^d,n\rangle}$.

The assumption for $\hat{\mathsf{s}}_{\langle p^d,n\rangle}$ is that any of the p bins are equally likely to have a tuple mapped to it. Thus, it is immaterial what the distribution of the attribute is, as long as our summary—the partitioning we have chosen—is an equi-width histogram of the data. This is a common way to summarize data in databases and data warehouses. Under these conditions, $\hat{\mathsf{s}}_{\langle p^d,n\rangle}$ is a good estimation.

4.2 Domain Distributions

In the basic model, we made no assumption about attributes' distributions, beyond the assumption of sparseness. Under the assumptions of sparseness and independence, the distributions are immaterial! This follows directly from Theorem 3.1 and its proof, and was illustrated in our discussion in Section 3. For one tuple to dominate another along a given dimension, the actual values do not matter, per se, just the tuples' *ranks* along that dimension.

Of course when the sparseness condition does not hold—and tuples tie on values—the skyline may no longer be independent of the attributes' distributions. In the previous section, we analyzed what happens when the sparseness assumption is relaxed, but we kept an assumption of uniform distributions. Now let us relax both the sparseness and the uniform distribution assumptions. Specifically, let us consider the domains' values under a Zipfian distribution, as this is a common distribution in real data.

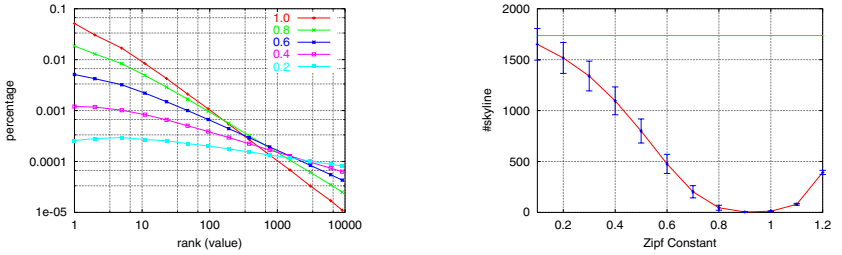
Definition 4.2. *Under a* Zipfian distribution *over p values $(1,\ldots,p)$ with* zeta constant *z, the probability of value i is $1/ci^z$ (where $c = \sum_{i=1}^{p} 1/i^z$, to normalize the probabilities). Zipfian distributions are also called zeta distributions. Usually z is close to unity.*

Let $\mathsf{s}_{\langle p_z^d,n\rangle}$ denote the number of skyline with respect to d min *criteria over n tuples for which the the values of each skyline attribute are distributed over p values $(1,\ldots,p)$ with a Zipfian distribution with zeta constant z, and the attributes are statistically independent.*

We ran Monte Carlo simulations as follows. For each *trial*, we generated one million tuples of d attributes randomly, each distributed with a Zipfian distribution across the integer range $1,\ldots,10,000$. A simulation then consisted of $10,000$ trials. To generate data following a Zipfian distribution, we employed the algorithm in [31]. Zipfian data when plotted on a log-log scale is linear. Figure 9(a) plots the generator's results for a million values at zeta values 1.0, .8, .6, .4, and .2, respectively, and demonstrates the fidelity of the generator, with the artifact of dropping off slightly near unity.[12]

The Zipfian distribution affects the likelihood of a tuple having a good score (near 1) on the dimension. The probability that the best possible tuple (all 1's) appears in the relation is increased, compared with the uniform distribution.

[12] We exponentially bin points together—1's, 2–3's, 4–7's, 8–15's, ...—to accommodate the log scale along the x-axis. For $z = 1.0$ we used $z = 1.001$ to avoid a singularity for $z = 1$ in the generator's algorithm.



(a) Column's values at different z.

(b) Plotting "$\hat{\mathsf{s}}_{\langle(10^4)_z^5\rangle,10^6}$".

Fig. 9. Skyline under Zipfian distributions.

In Figure 9(b), we plot simulations for zeta values from .1 to 1.2 in .1 steps. The results are quite reminiscent to those in Figure 8(a) for varying the number of values (p). To the right of the nadir are the cases when duplicates of the best tuple constitute the skyline. As before, these are Gaussian by the Central Limit Theorem, and $\mu = \sigma^2$. Consider the simulation with $z = .8$, near the nadir in Figure 9(b). As seen in Figure 9(a), the probability of value 1 when $z = .8$ is about .037. In a million draws (generated tuples), the probability of tuple $\langle 1, 1, 1, 1, 1\rangle$ occurring at least once is 7%. The distribution of $\mathsf{s}_{\langle(10^4)_8^5,10^6\rangle}$ is bimodal and is quite similar to that for $\mathsf{s}_{\langle 32^5,10^6\rangle}$ in Figure 8(b). To the left of the nadir, the distributions become Gaussian-like again and well-behaved.

The ceiling, indicated in Figure 9(b), on which $\hat{\mathsf{s}}_{\langle(10^4)_z^5,10^6\rangle}$ is asymptotically converging is $\hat{\mathsf{s}}_{\langle(10^4)^5,10^6\rangle}$ (1,737), the expected value for 10,000 partitions but with a uniform distribution. This ceiling is lower than $\hat{\mathsf{s}}_{5,10^6}$ (1,880). The reason is that as z decreases, the Zipfian distribution resembles more a uniform one.

When we consider skyline with respect to max criteria and Zipfian distributions, we see something different. In this case for $z = .5$ (and $d = 5$ and $n = 10^6$), our simulation yields 1,791, which is above the ceiling of 1,737. However, it is still below $\hat{\mathsf{s}}_{5,10^6}$. What happens now is that the higher values are rarer, and therefore are sparser. Most skyline tuples draw on these values (since max is the criterion), so it is as if the data is sparse. This will converge asymptotically from below on $\hat{\mathsf{s}}_{5,10^6}$ as z *increases*.

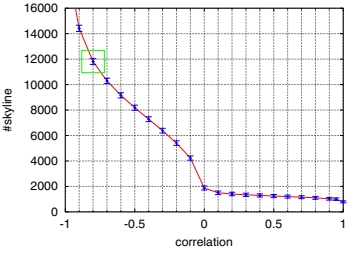
More generally, $\hat{\mathsf{s}}_{\langle p_z^d,n\rangle}$ converges asymptotically from below on $\hat{\mathsf{s}}_{\langle p^d,n\rangle}$ for min criteria, and on $\hat{\mathsf{s}}_{d,n}$ for max criteria. Our arguments extend for other distributions as well. Excluding the case of duplicate saturation, $\hat{\mathsf{s}}_{d,n}$ is a ceiling on the expected skyline cardinality.

4.3 Correlation and Anti-correlation

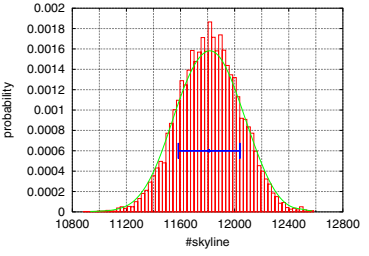
Our other assumption in the basic model in Definition 3.1 is that of statistical independence of the dimensions (skyline attributes). This is rarely true for real data, and skyline cardinality is sensitive to correlation.

Definition 4.3. *Let* $\mathsf{s}_{d,n}^{(r)}$ *denote the random variable that measures the number of skyline tuples of an* n*-tuple relation with respect to a* d*-dimensional skyline query, for which the* d *skyline attributes are statistically independent, save for one pair that are pair-wise correlated with correlation* r*.*

The skyline is affected quite differently by correlation and anti-correlation. High correlation between two skyline attributes acts as dimensional reduction. If tuple A has a better value on one of the dimensions than B, with high probability, it also does on the other dimension. At $r = 1.0$, that probability is one, so one of the dimensions is effectively eliminated. Anti-correlation is the antithesis of skyline, however. If A is better than B on one dimension, it is likely that B is better than A on the other. At $r = -1.0$, *all* tuples are in the skyline. In a way, skyline queries with highly anti-correlated skyline attributes do not make much sense. One is trying to optimize two values that are strictly trade-offs. Nevertheless, any realistic implementation of the skyline operator would have to accommodate such cases.



(a) Plotting "$\hat{\mathsf{s}}_{5,10^6}^{(r)}$".



(b) Distribution of $\mathsf{s}_{5,10^6}^{(-.8)}$.

Fig. 10. Skyline at different correlations.

To ascertain experimentally $\hat{\mathsf{s}}_{d,n}^{(r)}$ and the distribution of $\mathsf{s}_{d,n}^{(r)}$, as in Section 3, we ran 10,000 trials for each of $r = -0.9$ to 1.0 (and -0.95) by steps of 0.1 for $d = 5$ and $n = 10^6$. Figure 10(a) shows this. The error-bars represent the standard deviations. Figure 10(b) shows the distribution of $\mathsf{s}_{5,10^6}^{(-.8)}$, and is constructed in the same way as those in Figures 5(a), (b), and (c). The correlated cases behave as expected. Note that a simple linear interpolation between $d = 5$ and $d = 4$ would not be accurate to model the correlation. It would be easy though to fit a function for the interpolation. The $\hat{\mathsf{s}}$ do grow as anti-correlation increases, but not nearly as fast as expected. As anti-correlation increases, $\hat{\mathsf{s}}$ converges towards n very slowly. The distributions remain remarkably well behaved and Gaussian-like, and the standard deviations do not diverge rapidly.

We ought to understand further the effects of multiple correlations, and of group correlations (not just pair-wise), on skyline. To do so, it would be best

to have a model of multiple correlations that is compatible with the correlation information as kept by relational database systems. We plan to pursue this.

5 Ramifications and Conclusions

We have found the issues of skyline cardinality to be fascinating in their own right, but our primary goal in this endeavor has been to shed light on skyline queries, their effective use, and their efficient evaluation. Our analyses and insights should help us to understand better the uses of skyline queries, and help us in this next stage to build better, more robust algorithms for skyline's computation. Understanding skyline cardinality should give us and others better insights to devising good algorithms for skyline, and for comparing average-case performances of such algorithms.

We have found that the concrete estimate $\hat{\mathsf{s}}_{d,n}$ for skyline cardinality under our basic model is a *ceiling* on the cardinality when the model's assumptions are relaxed, save with two exceptions. This means we have a cardinality estimator for use in a relational query optimizer's cost model to accommodate skyline queries. That the concrete estimate $\hat{\mathsf{s}}_{d,n}$ is a ceiling is useful; over-estimations are better than under-estimations for the query optimizer. Query performance suffers more often when the optimizer's estimation is too low.

The two exceptions to the ceiling are the extreme degenerate case when the skyline simply consists of many copies of the same best tuple and the case of extreme anti-correlation. Both these cases are somewhat antithetical to the intention of skyline queries. The optimizer can be easily made to accommodate for the first case. It can price the query as if the skyline "preferences" were regular conditions *and* do a skyline cardinality estimation, and then pick the higher estimate. More is needed for the anti-correlation case, but we note that reasonable levels of anti-correlation are not as deadly for skyline as thought.

To be sure, work remains, and many issues to resolve. Golin [6] showed that changing the shape of the *space* of the points (tuples) can dramatically affect the skyline cardinality. For us, conditions prior to the skyline operator in a complex query or integrity constraints on the database could have this effect. This requires more study, and must be understood for skyline to be fully composable with other relational operators. We need a fuller understanding of skyline under correlation, for instance how multiple and group correlations affect the skyline, and how the attributes' distributions affect the skyline's cardinality when their domains are very small (that is, p is very small).

That $\hat{\mathsf{s}}$ *diminishes* in the presence of data denseness has ramifications for users of skyline queries—and preference queries more generally—not just for the cost model. A tempting strategy when the skyline query returns too few results for the user's liking is to bin the dimensions' values further. For instance, we might see little difference between a restaurant at which the average meal is \$24 and one at which it is \$21. So we might not want one restaurant trumping another on cost unless it were at least \$5 dollars less expensive. This could lead us to bin price into five dollar brackets. However, this reasonable-seeming strategy

backfires. As we have seen in Section 4.1, binning would only reduce further the number of choices (skyline answers).

Skyline queries offer a natural extension to min and max aggregation, and allow for one to query for nearest matches to one's objectives. The skyline operator, and potential extensions, may offer support for richer classes of queries with preferences. It may soon be worthwhile to add the skyline clause—and the underlying skyline operator—to relational systems.

We hope this work also yields more general insights into the nature of multidimensional data. In the long term, skyline may provide us, in turn, tools for the relational model and systems. For example, a skyline operator might provide the query optimizer with more efficient plans for evaluating queries with self-joins. Skyline statistics may provide a new type of useful database statistics that could yield better cost estimations for queries generally.

References

1. : Zagat Toronto Restaurants. Zagat Survey, LLC (2002)
2. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of the 17th ICDE. (2001) 421–430
3. Steuer, R.E.: Multiple Criteria Optimization: Theory, Computation, and Application. John Wiley & Sons, New York (1986)
4. Barndorff-Nielsen, O., Sobel, M.: On the distribution of the number of admissible points in a vector random sample. Theory of Probability and its Applications **11** (1966) 249–269
5. 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. Annals of Applied Probability **8** (1998) 886–895
6. Golin, M.J.: Maxima in convex regions. In: Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms (SODA), ACM/SIAM (1993) 352–360
7. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. JACM **22** (1975) 469–476
8. Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Springer-Verlag (1985)
9. Bentley, J.L., Kung, H.T., Schkolnick, M., Thompson, C.D.: On the average number of maxima in a set of vectors and applications. JACM **25** (1978) 536–543
10. Buchta, C.: On the average number of maxima in a set of vectors. Information Processing Letters **33** (1989) 63–65
11. Bentley, J.L., Clarkson, K.L., Levine, D.B.: Fast linear expected-time algorithms for computing maxima and convex hulls. In: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), ACM/SIAM (1990) 179–187
12. Matoušek, J.: Computing dominances in E^n. Information Processing Letters **38** (1991) 277–278
13. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: Proceedings of the 19th International Conference on Data Engineering (ICDE). (2003)
14. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: Proceedings of the 28*th* Conference on Very Large Databases (VLDB). (2002)

15. 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, ACM Press (2003) To appear.
16. Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient progressive skyline computation. In: Proc. of 27th VLDB. (2001) 301–310
17. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In Carey, M.J., Schneider, D.A., eds.: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, ACM Press (1995) 71–79
18. Berchtold, S., Böhm, C., Keim, D.A., Kriegel, H.P.: A cost model for nearest neighbor search in high-dimensional data space. In: Proceedings of the Sixteenth PODS. (1997) 78–86
19. Katayama, N., Satoh, S.: The SR-tree: An index structure for high-dimensional nearest neighbor queries. In: Proceedings of Sigmod. (1997) 369–380
20. Chomicki, J.: Querying with intrinsic preferences. In: Proceedings of EDBT. (2002)
21. Chu, W.W., Yang, H., Chiang, K., Minock, M., Chow, G., Larson, C.: CoBase: A scalable and extensible cooperative information system. Journal of Intelligent Information Systems (JIIS) **6** (1996) 223–259
22. Gaasterland, T., Godfrey, P., Minker, J.: Relaxation as a platform for cooperative answering. Journal of Intelligent Information Systems (JIIS) **1** (1992) 293–321
23. Gaasterland, T., Lobo, J.: Qualifying answers according to user needs and preferences. Fundamenta Informaticæ **32** (1997) 121–137
24. Minker, J.: An overview of cooperative answering in databases. In Andreasen, T., Christiansen, H., Larsen, H.L., eds.: Third International Conference on Flexible Query Answering Systems (FQAS'98). (1998) 282–285
25. Agrawal, R., Wimmers, E.L.: A framework for expressing and combining preferences. In: Proc. of Sigmod. (2000) 297–306
26. Hristidis, V., Koudas, N., Papakonstantinou, Y.: Prefer: A system for the efficient execution of multi-parametric ranked queries. In: Proceedings of Sigmod. (2001) 259–270
27. Kießling, W.: Foundations of preferences in database systems. In: Proceedings of the 28*th* Conference on Very Large Databases (VLDB). (2002)
28. Kießling, W., Köstler, G.: Preference SQL: Design, implementation, experiences. In: Proceedings of the 28*th* Conference on Very Large Databases (VLDB). (2002)
29. Roman, S.: The logarithmic binomial formula. American Mathematics Monthly **99** (1992) 641–648
30. Knuth, D.E.: Fundamental Algorithms: The Art of Computer Programming. second edn. Volume 1. Addison Wesley, Reading, MA (1973)
31. Gray, J., Sundaresan, P., Englert, S., Baclawski, K., Weinberger, P.J.: Quickly generating billion-record synthetic databases. In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, ACM Press (1994)