SkyCover: Finding Range-Constrained Approximate Skylines with Bounded Quality Guarantees

Shubhendu Aggarwal
shubhu@cse.iitk.ac.in

Shubhadip Mitra
smitr@cse.iitk.ac.in

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

Dept. of Computer Science and Engineering,
Indian Institute of Technology, Kanpur.
India

ABSTRACT

Skyline queries retrieve promising data objects that are not dominated in all the attributes of interest. However, in many cases, a user may not be interested in a skyline set computed over the entire dataset, but rather over a *specified range of values* for each attribute. For example, a user may look for hotels only within a specified budget and/or in a particular area in the city. This leads to *constrained* skylines. Even after constraining the query ranges, the size of the skyline set can be impractically large, thereby necessitating the need for *approximate* or *representative* skylines. Thus, in this paper, we introduce the problem of finding *range-constrained approximate* skylines. We design a grid-based framework, called SkyCover, for computing such skylines. Given an approximation error parameter $\epsilon > 0$, the SkyCover framework guarantees that every skyline is "covered" by at least one representative object that is not worse by more than a factor of $(1 + \epsilon)$ in all the dimensions. This is achieved by employing a non-uniform grid partitioning on the data space. We also propose two *new* metrics based on the covering factor to assess the quality of an approximate skyline set. Experimental evaluation reveals that SkyCover outperforms the competing methods in both quality and running time.

1. INTRODUCTION

Since their introduction to the database community by [3], skyline queries have attracted a significant interest from researchers, and have also found their way into commercial databases such as PostgreSQL [11]. Skylines are especially suitable for situations where there are multiple attributes of interest but no clear optimization function that can help choose a single preferred object. The skyline set returns all promising objects that are *not worse* than another object in *all* the attributes. There is typically no ordering among the skyline objects.

A preference function needs to be specified for every dimension of interest for the skyline query. For example, while looking for good hotel deals online, the attributes of interest may be cost and distance to city center, and the preference functions are *less than* for both. A user *never* chooses a hotel H_1 that costs more *and* is farther from city center than another hotel H_2. The hotel H_1 is, hence, not a skyline.

Formally, assume that there is a dataset D with d skyline attributes having the preference function in each attribute to be less than ($<$).[1] In other words, for every skyline dimension i, a smaller value dominates a larger one. A point t *dominates* another point u, denoted by $t \succ u$, if and only if $\forall i,\ t_i \leq u_i$ and $\exists j,\ t_j < u_j$. A point t is in the *skyline* set $S \subseteq D$ if and only if there does not exist any other point u in the dataset that dominates it, i.e., $t \in S \iff \nexists u \in D$ such that $u \succ t$.

Note that in the definition of skylines, the *exact* values of the objects are not important; only the relative ordering matters. Thus, the ranges of the values can be modified (such as scaling and shifting) as long as the preference relationships are maintained.

When the number of dimensions is large, the skyline query suffers from the cardinality problem in the sense that the size of the skyline set may be too large. Since one of the basic utilities of the skyline query is to deduce the set of useful objects by discarding the useless ones from the dataset that cannot be good, the entire exercise of finding skylines may suffer.

In many cases, a user may not be interested in all the objects over the entire range of the dataset. For example, while searching for hotels, the user may have a certain budget and/or a particular area of the city in mind. Hence, the skyline query needs to be appropriately modified as well. The query should return the skyline set considering only the specified range as the dataset. Informally, we refer to such skyline objects over a specified query range as *range-constrained skyline* objects.

Given a query range Ω, specified by d ranges of the form $[l_i, u_i)$ for $i = 1, \ldots, d$, let $D(\Omega)$ denote the induced set of objects (in D) that fall within Ω. An object t is referred to as a *range-constrained skyline* of $D(\Omega)$ if there does not exist any object $s \in D(\Omega)$ that dominates it. In other words, t is a skyline object of the set $D(\Omega)$. Note that t may not be a skyline object in D, as it may be dominated by some object not in $D(\Omega)$. Therefore, the range-constrained skylines cannot be simply computed from the skyline set of D by applying the range Ω.

Although the range-constrained skylines have been addressed in the literature [10, 17], they still do not address the cardinality issue satisfactorily. For high dimensions, even the range-constrained skylines can be very large in number. The cardinality may be prohibitively large also when the query range Ω is large.

To address such high number of range-constrained skylines, in this work, we propose *approximate range-constrained* skyline computation. As there are several existing works on *approximate* or

[1] A greater than preference function can always be converted by inverting or negating the values.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 21st International Conference on Management of Data.
COMAD, March 11-13, 2016, Pune.
Copyright 2016 Computer Society of India (CSI).

representative set of skylines [6,23,32], a natural proposition would be to employ one of these techniques to compute approximate range-constrained skylines. Given a query range Ω, a range query is first performed over D to compute the induced dataset $D(\Omega)$. Following this step, skylines $S(\Omega)$ are computed over the set of objects $D(\Omega)$. Finally, approximate skylines are computed from $S(\Omega)$.

However, this strategy has the following limitations:

- **Low performance:** The approximation is performed only *after* computing the skyline set which needs the range query to be solved before that. Thus, this can be fairly time consuming, especially when the range query answer set and the skyline set are large.
- **Low quality:** Most of the existing approximate skyline computation techniques do not cover the skyline set, i.e., for every skyline object in the original dataset, there may not exist a suitable representative object in the reported set. Moreover, they consider optimizing metrics such as k most diverse skylines or k most dominating skylines which are NP-hard problems and, therefore, resort to heuristics that yield sub-optimal solutions.
- **High storage:** The storage of the sets $D(\Omega)$ and $S(\Omega)$ may be fairly large.

Motivated by these challenges, in this paper, we propose a new indexing framework called SKYCOVER that reports a set of approximate range-constrained skylines. The approximation is pushed *within* the skyline operator and not afterwards. Further, the approximation is achieved by using a grid structure that *guarantees* the coverage up to a user-defined error threshold. In other words, the reported representative data objects *cover* the *entire* set of range-constrained skylines with bounded approximation errors.

The SkyCover framework employs a hashing scheme that ensures the bounded quality guarantees, without requiring any separate approximate skyline technique. The proposed index structure stores a sampled set of objects that is sufficient to offer the quality guarantees. As a result, the number of input objects to the skyline finding routine is reduced, thereby making it faster. The set returned by the skyline query on this reduced input set is reported as the SkyCover set, which is an approximate and succinct representation of the range-constrained skyline set. In addition to its efficiency, the proposed SkyCover framework is simple and easily extensible to several practical variations of skyline queries, such as top-k skyline queries [4, 31], nearest-neighbor skyline queries [19], progressive skyline queries [24, 29], skyline queries over dynamic and streaming settings [22, 28], etc.

The basic idea in SkyCover is that if there are two or more objects that are quite similar to each other, the information added beyond the first object is low, and therefore, the subsequent objects can be filtered out. The similarity is based on the proximity of the values in the underlying data space. For all such objects that have values close to each other, the SkyCover index structure retains only one of them.

Ideally, the value of a skyline object that is not reported should be within some threshold of tolerance within the skyline object that is reported. The approximation error is captured by an error parameter ϵ. SkyCover guarantees that every range-constrained skyline object is *covered* by at least one object in the SkyCover set that is not worse by a factor of $(1 + \epsilon)$ in all the dimensions. Such an approximation factor is natural in many applications. For example, when a skyline hotel H_1 has no vacancies, a user may tolerate a hotel H_2 that has larger cost and distance than H_1 within a factor of, say, 5%. This translates to finding H_2 with cost less than or equal to 105 units and distance less than 2.1 units when the H_1 has a cost of 100 units and distance 2 units. The value of $\epsilon = 0.05$ guarantees this.

In sum, our contributions in this paper are as follows:

1. To the best of our knowledge, this is the first work to directly address the problem of approximate range-constrained skyline computation.
2. We propose a novel SkyCover framework for efficiently computing approximate range-constrained skylines with bounded quality guarantees.
3. We propose two new metrics for measuring the quality of an approximate skyline set based on the concept of *covering factor*.
4. Experimental evaluation, on both real and synthetic datasets, reveals that SkyCover outperforms the other competing methods in both quality and running time.

The outline of the rest of the paper is as follows. Section 2 discusses the related work. Section 3 describes the SkyCover framework, the properties of which are analyzed in Section 4. In Section 5, we propose two new quality metrics. The results are described in Section 6. Section 7 concludes.

2. RELATED WORK

We first discuss the works in the area of skyline computation and then describe the related literature for range-constrained skylines and approximate skylines.

2.1 Skylines

The skyline problem has been first studied as that of determining the maxima of a set of vectors. The algorithms in this field are typically based on the divide-and-conquer [20] and parallel approaches [25]. These methods assume that the entire data fits in the main memory.

The notion of skyline for relational database systems was introduced by [3] who proposed the "skyline" operator. Since then, a large number of algorithms, both index-based and non-index-based, have been proposed for skyline computation. Index-based methods include B-tree based schemes [3], bitmap and index [29], nearest-neighbor (NN) [19] and branch-and-bound (BBS) [24]. More recently, a trie-based index structure has been proposed in [27] to improve scalability and maintenance costs in case of data updates. In non-index-based methods, block-nested-loop (BNL) and divide-and-conquer (DC) were proposed in [3]. Chomicki et al. [7] proposed a variant of BNL algorithm called the sort-filter-skyline (SFS) algorithm that involves pre-processing of sorting the data with respect to a monotone scoring function.

For high dimensional data, there has been a lot of focus on reducing the number of skylines reported. Since it becomes increasingly difficult for a point to dominate another in high dimensions, k-dominance [6] was introduced to relax the dominance criteria where it is sufficient to be better in k dimensions in order to dominate another point.

The strategy of grid-based partitioning has been used in the distributed environment [1, 8, 21, 26, 33]. Essentially, in these works, a grid is created on the data such that each grid cell has almost the same amount of data. The same grid structure is imposed by all the distributed nodes on the data space. Each node computes the skylines in a specific grid (or a set of grids), and finally the local results are merged to compute the desired skyline set.

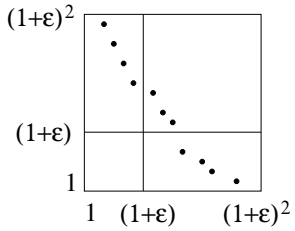


Figure 1: Example of ϵ-SkyCover.

2.2 Range-Constrained Skylines

The problem of constrained subspace skyline computation was introduced in [10]. In order to support constrained skylines for arbitrary subspaces, they presented approaches exploiting multiple low-dimensional indexes instead of a single high-dimensional index. Range-constrained skylines in two dimensions was investigated in [17]. None of these works address the issue of high cardinality of the skyline set.

2.3 Approximate Skylines

There are several works on approximate or representative skylines. One one hand, [9, 13, 23] focused on finding k skyline points that together dominate the maximum number of non-skyline points. On the other hand, [14, 30] considered returning k skyline points that best represents the skyline contour. Other works such as [32] focused on reporting the k most diverse skyline points, where the diversity is measured using Jaccard distance.

Approximately dominating representatives (ADRs) [18] reports ϵ-approximate skyline for some ϵ given by the user. An ϵ-ADR query returns a set of points which together when boosted by ϵ in all dimensions dominate all other points (they considered greater than $>$ as the preference function for all dimensions). This is similar to the notion we implement as well. However, their work is significantly different from ours as they assume that the skyline set has been pre-computed and the task is to post-process and determine the smallest possible ADR. We, on the other hand, push the approximation scheme before the skyline operation. They also show that for a given ϵ, the problem of minimizing the cardinality of ϵ-ADR is NP-hard for three or more dimensions.

Approximate skylines over a data stream was computed in [28] which gave a notion of additive errors in skylines by defining strong domination. A point t strongly dominates another point u if for all dimensions i, $t_i + \epsilon \leq u_i$ where ϵ is an acceptable difference distance. Their algorithm guarantees that any two approximate skyline points are at least 2ϵ apart and for every skyline point there is an approximate skyline point within ϵ distance of it. An ϵ-skyline was proposed in [34] where the domination criteria called ϵ-domination was modified to allow additive error of ϵ along each weighted dimension. Thick skylines were proposed in [16] where all skyline points and their ϵ-neighbors (points within ϵ distance) are reported. The algorithm involves computing distances between points which is expensive.

2.4 Comparison with SkyCover

Next we closely compare our SkyCover approach with the k-RSP approach [23] and SkyDiver approach [32], as they are the most popular notions of representative skylines. For experimental evaluation, we compare the quality of the solutions realized by SkyCover against these two approaches.

While all the three approaches attempt to suitably represent the skyline set, the computational challenge in optimally (and also approximately) computing these solutions vary significantly. While the sets reported by k-RSP and SkyDiver approaches are of fixed cardinality of size k, the size of the set returned by our SkyCover approach is not fixed. Due to this reason, while the optimal computation of the representative sets in case of k-RSP and SkyDiver is NP-hard, the computation of the representative set by our SkyCover approach is polynomially solvable. While our approach guarantees that the reported representative points have bounded approximation errors in all dimensions w.r.t. the skylines missed (not reported), there is no such guarantee in the other two approaches. For instance, consider the example shown in Figure 1. Observe that each point in the dataset is a skyline. Suppose a user wants $k = 3$ representative points. Since the dominating sets, i.e., the set of points dominated by the given point, for each of the points is empty, both the k-RSP and the SkyDiver approaches report any 3 points arbitrarily. Therefore, the worst case approximation factor of the representative sets returned by these two approaches can be arbitrarily bad. On the other hand, as it will be described later, our ϵ-SkyCover would select one point from each of the three grid cells containing the points, thereby guaranteeing that every missed skyline has a representative point that is worse by a multiplicative factor of at most $(1 + \epsilon)$.

The k-RSP and SkyDiver problems being NP-hard for dimensions greater than two, it was shown that when greedy heuristic is applied to them, they can be approximated within a factor of $1 - \frac{1}{e}$ and 2 respectively. For scalability and efficiency reasons, each of these heuristics consider randomized techniques that offer theoretical accuracy guarantees. While k-RSP approach employ probabilistic counting using FM algorithm [12], the SkyDiver employs min-hash (MH) [5] and LSH [15]. In contrast, we propose an efficient deterministic algorithm based on the concept of hashing using non-uniform grid-based partitioning, whose complexity depends on the desired value of the approximation parameter.

Most importantly, the k-RSP and the SkyDiver approaches can be applied only *after* the skyline computation which is quite expensive. Our SkyCover approach is applied *prior* to skyline computation, which reduces the size of the dataset considerably, thereby leading to savings in query-time.

In addition, our SkyCover framework is more generic as any of the post-processing approximation techniques including k-RSP and SkyDiver techniques can be applied after the approximation and skyline computations have been done. Hence, our work can be treated both as alternative as well as complementary to the existing works in the space of approximate skyline representation.

3. THE SKYCOVER FRAMEWORK

In this section, we first introduce the problem of approximate range-constrained skyline computation. Then, we describe the proposed SkyCover framework for finding such skylines. Finally, we analyze its running time, and discuss possible extensions of the framework.

3.1 Range-Constrained Skylines

Assume that the dataset D is d-dimensional with the range in each dimension between 1 and R. The total data space, therefore, is $[1, R]^d$. Suppose S is the skyline set of D. All the symbols used in this paper are listed in Table 1 for easy reference.

We first introduce the notion of range-constrained skylines. Given a range $\Omega = \prod_i [l_i, u_i)$, where $(1 \leq l_i \leq u_i \leq R)$, $D(\Omega)$ denotes the induced dataset over Ω, i.e., it contains the points in D that lie within the range Ω. Hence, $p \in D(\Omega)$ if and only if $l_i \leq p_i \leq u_i \forall i$. The skylines $S(\Omega)$ computed over $D(\Omega)$ is the range-constrained skyline set.

Table 1: List of Symbols.

Symbol	Description
D	Dataset
$N = \|D\|$	Cardinality of dataset
d	Dimensionality
$[1, R]$	Range of values along a dimension
S	Skyline set
$s = \|S\|$	Cardinality of skyline set
$\Omega = \prod_i [l_i, u_i)$	Query range
$D(\Omega)$	Induced dataset over the range Ω
$n = \|D(\Omega)\|$	Cardinality of induced dataset
$S(\Omega)$	Range-constrained skyline set of $D(\Omega)$
ϵ	Error parameter
$\Omega_\epsilon = [l'_i, u'_i)$	Stretched range of Ω
$l'_i = l_i/(1+\epsilon)$	Lower range of Ω_ϵ
$u'_i = u_i.(1+\epsilon)$	Upper range of Ω_ϵ
$S'(\Omega_\epsilon)$	SkyCover set of $D(\Omega)$
g	Number of grid partitions along a dimension
$P(\Omega)$	Set of grid representatives that lie in Ω_ϵ
$m = \|P(\Omega)\|$	Total number of grid representatives stored
S'	SkyCover set of D
$k = \|S'(\Omega_\epsilon)\|$	Cardinality of SkyCover set

DEFINITION 1 (RANGE-CONSTRAINED SKYLINE). *Given a range Ω, a point $s \in D(\Omega)$ is in the* range-constrained skyline *set $S(\Omega)$ of $D(\Omega)$ if and only if it is not dominated by any other point in $D(\Omega)$.*

It is important to note that $S(\Omega)$ is not necessarily a subset of the skyline set S, as it may contain a point that is dominated by another in D but not in $D(\Omega)$. A range-constrained skyline query is, thus, a generalization of the regular skyline query.

Although by constraining the data range, the size of the dataset is reduced, there is no guarantee that the size of the skyline set will be reduced. This may happen since there may be a globally dominating skyline outside the range, and not considering it allows many other objects as skylines.

Even otherwise, the number of range-constrained skylines may be too many, especially for high-dimensional spaces. Hence, a flavor of representation or approximation is needed to reduce the number. For that, we introduce the concept of skyline cover next.

3.2 Skyline Cover

Given an $\epsilon > 0$, we first define ϵ-cover.

DEFINITION 2 (ϵ-COVER). *Given $\epsilon > 0$, a point $t \in D$ ϵ-covers a point $s \in D$, if and only if for every dimension i, t is not worse than s by a multiplicative factor of $(1+\epsilon)$*

$$t \text{ } \epsilon\text{-covers } s \iff \forall i,\ t_i \leq (1+\epsilon)s_i \tag{1}$$

The simple dominance is a special case of ϵ-cover when $\epsilon = 0$.

Using this, we next define the concept of ϵ-SkyCover. Informally, a set $S'(\Omega_\epsilon)$ is a *skyline cover* of $S(\Omega)$ if each point in $S(\Omega)$ has a suitable representative in $S'(\Omega_\epsilon)$. The skyline cover $S'(\Omega_\epsilon)$ is an ϵ-SkyCover if every point $s \in S(\Omega)$ is represented within the multiplicative factor of $(1+\epsilon)$, i.e., it is ϵ-covered by at least one point $t \in S'(\Omega_\epsilon)$.

DEFINITION 3 (ϵ-SKYCOVER). *A set $S'(\Omega_\epsilon)$ is an ϵ-*SkyCover *of $S(\Omega)$ if every point in $S(\Omega)$ is ϵ-covered by at least one point in $S'(\Omega_\epsilon)$:*

$$S'(\Omega_\epsilon) \text{ is an } \epsilon\text{-SkyCover of } S(\Omega) \iff \forall s \in S(\Omega), \exists t \in S'(\Omega_\epsilon) \text{ s.t. } t \text{ } \epsilon\text{-covers } s \tag{2}$$

Assume that Ω_ϵ denotes the range of Ω that is stretched by a factor of $(1+\epsilon)$ along each dimension, i.e., $\Omega_\epsilon = \prod_i [l'_i, u'_i) \forall i$ where $l'_i = l_i/(1+\epsilon)$ and $u'_i = u_i.(1+\epsilon)$. The skyline set $S'(\Omega_\epsilon)$ of $D(\Omega_\epsilon)$ is *necessarily* an ϵ-SkyCover of the skyline set $S(\Omega)$ of $D(\Omega)$.

When the context is clear, we refer to an ϵ-SkyCover by simply SkyCover.

The parameter ϵ essentially captures the user's tolerance to approximation. For example, consider the cost attribute of a particular dataset. If the cost of a skyline point is 100 units and $\epsilon = 0.05$, it means that the user can tolerate a point that costs up to 105 units.

Note that ϵ is a multiplicative ratio which makes more sense than an absolute margin (i.e., additive error) such as 5 units in this context. When the cost of an object is more, the difference in cost is more as well. When it is less, the difference automatically shrinks. Hence, in the above example, if a skyline object costs 10 units, it is more likely that a user will tolerate up to 10.5 units and not 15 units. Similarly, if the cost is 1000 units, the user can go up to 1050 units and not 1005 units.

3.3 Problem Statement

In this paper, we address the following query:

PROBLEM 1. *Given a range $\Omega = \prod_i [l_i, u_i)$, where $(1 \leq l_i \leq u_i \leq R)$, report its SkyCover, i.e., $S'(\Omega_\epsilon)$.*

Note that the range Ω is available only at query time. Therefore, since $S'(\Omega_\epsilon)$ need not be a subset of S or its SkyCover S', even if we pre-compute the sets S or S', it is not easy or efficient to compute the set $S'(\Omega_\epsilon)$ using these pre-computed sets.

The goal of the proposed SkyCover framework is to, thus, efficiently compute the SkyCover $S'(\Omega_\epsilon)$. To achieve this, we first build an index structure using sampled data points in D. On receiving the query range Ω, we identify a set of points stored in the index structure that lie in the stretched range of Ω, i.e., Ω_ϵ. Finally, we compute the skylines over this set of points to return the desired SkyCover. We next discuss these steps in more detail.

3.4 Grid Partitioning

We construct a grid-based index structure by employing a non-uniform grid partitioning scheme. The grid boundaries are imposed at multiplicative intervals of $(1+\epsilon)$. Thus, the first grid cell is from $(1+\epsilon)^0 = 1$ to $(1+\epsilon)^1$, the second one from $(1+\epsilon)^1$ to $(1+\epsilon)^2$, and so on. The number of grid partitions, g, along any dimension is, therefore, equal to

$$g = \lceil \log_{1+\epsilon} R \rceil \tag{3}$$

Hence, the total number of grid cells for d-dimensional space is g^d.

Assume that $R = (1+\epsilon)^g$ such that there are exactly g grid partitions along any dimension. (The data space can always be stretched so that this happens.) Otherwise, using the ceiling function for g imposes an error parameter $\epsilon' < \epsilon$, thus, protecting the $(1+\epsilon)$ SkyCover guarantees.

If the range of a dimension is $[min, max]$, it can always be shifted and stretched to fit $[1, R]$. If $min = 1$, then the ratios after the transformation do not change. Otherwise, the error value ϵ in the $[min, max]$ range can be mapped to an ϵ' in the $[1, R]$ range if $min > 0$. If both min and max are negative, the ratios are inverted. If, however, $min < 0$ and $max > 0$, then the

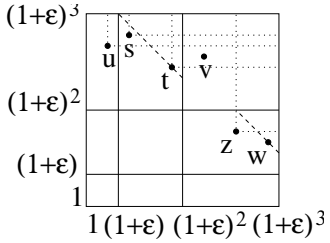


Figure 2: Grid-based partitioning.

concept of multiplicative error makes little sense and we do not consider such cases. As explained in Section 3.10, an additive error is more meaningful there and the SkyCover framework can be appropriately modified to handle that using uniform grid partitions.

A grid cell $x = (x_0, \ldots, x_{d-1})$ is indexed by the corresponding d grid cell numbers along the dimensions. Here, $0 \leq x_i \leq g-1$ denotes the grid cell index for the i^{th} dimension.

A point $t = (t_0, \ldots, t_{d-1})$ falls in grid cell $x = (x_0, \ldots, x_{d-1})$ if and only if it satisfies the condition $\forall i,\ (1+\epsilon)^{x_i} \leq t_i < (1+\epsilon)^{x_i+1}$. The hash function $h_i(t_i)$ that returns the grid cell index for the attribute i of a point t is, thus, $h_i(t_i) = \lfloor \log_{(1+\epsilon)} t_i \rfloor$. The hash function $h(t)$ for a point t is the combination of the corresponding hash functions for each dimension:

$$h(t) = \langle h_0(t_0), \ldots, h_{d-1}(t_{d-1}) \rangle \quad (4)$$

Figure 2 shows an example partitioning of the space with $g = 3$ and $d = 2$.

Note that the grids are imposed on the original $[1, R)^d$ space before the query range is made available at run time.

3.5 Choosing Grid Cell Representative

All the points that hash to the same grid cell are represented by only one of them. In this way, the SkyCover stores a much smaller sample of the dataset. Since any point within the cell ϵ-covers any other point within the same cell, keeping anyone of them would serve the purpose of ϵ-SkyCover (shown in Theorem 1). Thus, this not only ensures quality guarantees, but also significantly contributes towards lower storage and, therefore, higher efficiency.

We next discuss how to choose the cell representative. For each point t, we employ an entropy function similar to that used in the SFS algorithm [7]: $ent(t) = \sum_{i=0}^{d-1} t_i$. Since the ultimate goal is to compute range-constrained skylines, the entropy function is re-used later during the actual skyline computation phase.

The grid representative of a cell x is chosen as the point t with the lowest entropy: $t = \arg\min\{ent(t) | t \in x\}$. Note that no other point in the same cell can dominate t as then its entropy would have necessarily lower. Moreover, the scheme guarantees that t ϵ-covers any other point in the same cell.

3.6 Query Processing

When the query parameters Ω and ϵ arrive, the first step is the retrieval of the set of grid representatives $P(\Omega)$ that map to the *stretched* range of Ω, i.e., Ω_ϵ. Assume the query range is $\Omega = \prod_i [l_i, u_i)$. Along dimension i, we identify the grid cell indices that map to the above space as follows. Assume $x_i = \lfloor \log_{1+\epsilon} l_i \rfloor$ and $y_i = \lfloor \log_{1+\epsilon} u_i \rfloor$. The set $P(\Omega)$ denotes the set of all the grid representatives of the grids that lie in the space $\prod_{i=0}^{d-1} [x_i, y_i]$:

$$P(\Omega) = \{\text{grid representative of cell } z |\ \forall i,\ x_i \leq z_i \leq y_i\} \quad (5)$$

Next, a skyline finding algorithm is employed over the set of points in $P(\Omega)$. The answer thus obtained is reported as the SkyCover $S'(\Omega_\epsilon)$.

Algorithm 1 SkyCover Algorithm

```
1: procedure OFFLINE COMPUTATION (Dataset D, Error parameter ε)
2:     H ← Φ
3:     for all t ∈ D do
4:         h(t) ← grid index of t using ε in Eq. (4)
5:         if h(t) is empty then
6:             Insert h(t), ⟨t, ent(t)⟩ in H
7:         else
8:             ⟨u, ent(u)⟩ ← value(h(t))
9:             if ent(t) < ent(u) then
10:                value(h(t)) ← ⟨t, ent(t)⟩
11:            end if
12:        end if
13:    end for
14:    return H
15: end procedure
1: procedure ONLINE COMPUTATION (Query range Ω)
2:     Compute P(Ω) from H as described in Section 3.6
3:     return S'(Ω_ε) ← FINDSKYLINES(P(Ω))
4: end procedure
```

For our implementation, we choose the SFS algorithm [7] as it is an online algorithm with no requirement of index construction and, is, thus, applicable in all situations. Moreover, it is quite simple, easy to implement, and fairly efficient in terms of running time.

3.7 The SkyCover Algorithm

Using the above ideas, we now summarize the steps in the SkyCover framework (pseudocode shown in Algorithm 1).

The framework comprises of two phases, the offline phase and the online phase. During the offline phase, we build the SkyCover index structure. A dynamic hash table H is maintained with the grid cell index as the key and the 2-tuple consisting of the representative point and its entropy, as the value. The data points are processed one at a time. For a point t, its grid cell $h(t)$ is computed using Eq. (4). If there is no such key in H, a key-value pair with key $h(t)$ and value $\langle t, ent(t) \rangle$ is inserted. If, however, there is already a key, then the value corresponding to it is extracted. If the entropy of the new point is less than that of the old one, the old value is replaced with the new point (and its entropy). Otherwise, since there is already a representative of this cell with a lower entropy, the new point is discarded.

The online phase is discussed in Section 3.6.

3.8 Analysis of Running Time

In this section, we compare the running time of SkyCover algorithm with that of the naïve approach, discussed in Section 1, that requires range search, followed by skyline computation, and finally approximate skyline computation.

Assume that the cardinality of the dataset D is N, the dimensionality is d, the size of the skyline set S is s, the number of representative points left using the grid-based partitioning is m, and the size of the SkyCover set extracted from m is k.

We assume that if the cardinality of the induced dataset $D(\Omega)$ is n, the range search required to find all the points in the range Ω is at least $O(n)$.

We first analyze the naïve approach. For skyline computation, we consider the SFS algorithm. The SFS algorithm has the following costs: (i) Computing entropy: $O(nd)$ for mapping d dimensions to a single value for all n points, (ii) Sorting the points based

on entropy: $O(n \log n)$, (iii) Skyline computation: $O(ns)$ assuming a window size of $O(s)$. Hence the total time complexity of the SFS algorithm is $O(n.d + n \log n + n.s)$.

Finally, using any existing approximate skyline techniques [6, 23, 32], reporting k representative skylines would require at least $O(sk)$ time. Hence the total time complexity of the naïve approach is at least$O(nd + n \log n + n.s + s.k)$.

The running time of the SkyCover framework comprises of the following components: (i) Computing entropy and hashing during the offline phase: $O(nd)$ assuming $O(1)$ hashing costs, (ii) Retrieving all the hash values during online phase: $O(m)$, (iii) Sorting based on entropy (during online phase): $O(m \log m)$, (iv) Skyline computation during online phase: $O(mk)$. Thus, the offline cost is $O(n.d)$, and the online cost is $O(m \log m + m.k)$.

It was shown in [2], that if the attribute values are chosen independently, then the average number of skylines is $O((\ln n)^{d-1})$. Thus, assuming $s = O((\log n)^{d-1})$, $k = O((\log m)^{d-1})$ and $d = O(1)$, and considering only the dominant terms, the query time of the naïve approach simplifies to $O(n.(\log n)^{d-1} + (\log m)^{d-1}.(\log n)^{d-1})$, and that of the SkyCover becomes $O(m.(\log m)^{d-1})$. As $m < n$, the running cost of SkyCover framework is always better than that of the naïve approach, at least by an additive factor of $O((\log m \log n)^{d-1})$. In Section 4.3, we present an analysis of expected value of m.

3.9 SkyCover using Uniform Grids

The SkyCover framework can be easily modified to work with uniform grid partitions. In this scheme, the total range $R - 1$ in each dimension is split into g equal parts. The grid boundaries for any dimension, therefore, fall at $1, 1 + (R-1)/g, 1 + 2(R-1)/g$, etc. The total number of cells is again g^d, i.e., the same as non-uniform partitioning. The SkyCover framework remains the same for the two cases.

3.10 Extensions of SkyCover

The grid-based hashing mechanism employed in the SkyCover framework is generic enough to capture different kinds of errors. For example, the error parameter can be different for each dimension. The number of partitions, g_i, for dimension i, is computed using the corresponding error parameter ϵ_i.

More importantly, SkyCover is not restricted to only multiplicative errors. For example, guarantees for additive errors can be easily offered using uniform grid partitions. If the additive error tolerance is δ, the width of the grid cells should be at most δ.

Moreover, the SkyCover framework allows seamless integration of skyline queries over attributes demanding different types of error tolerance. While some attributes may tolerate multiplicative errors, some others may require additive errors, and the rest may not tolerate any error. The dimensions for multiplicative error attributes may be partitioned in a non-uniform manner, while those for additive errors may be partitioned using equal grid widths. The dimensions that do not tolerate any error will not be hashed at all using the grid partitions.

We have, however, stuck to the multiplicative SkyCover framework in this paper and have postponed the detailed experimentation and analysis of the generalized framework to a later paper.

We also claim that the SkyCover framework is applicable to streaming data settings [28] with insert-only operations. The multi-dimensional data points arriving in the stream can be efficiently hashed into the SkyCover index structure, and subsequently queried upon. Similarly, the framework is particularly suited for update-heavy workloads where most of the small updates in the values of a point can be absorbed since it lies within the same cell. The update in the skyline set needs to be checked only when a point moves to a cell that was not occupied earlier. In addition, we claim that the framework can be easily adapted for other settings such as distributed environments [26], and moreover, help in computing approximate top-k skylines [4, 31], progressive skylines [24, 29], etc. However, in this paper, we do not evaluate the performance of SkyCover over such settings.

4. PROPERTIES OF SKYCOVER

In this section, we first discuss the correctness of the SkyCover framework, followed by quality guarantees on falsely reported range-constrained skylines. Finally, we present a thorough analysis of the expected number of grid representatives, m.

4.1 Correctness

The following theorem establishes the correctness.

THEOREM 1. *Given any range Ω, the SkyCover framework correctly computes its ϵ-SkyCover, i.e., (1) $S'(\Omega_\epsilon) \subseteq D(\Omega_\epsilon)$, and (2) for every range-constrained skyline point $s \in S(\Omega)$, there exists at least one $t \in S'(\Omega_\epsilon)$ that ϵ-covers s.*

PROOF. Assume the query range to be $\Omega = \prod_i [l_i, u_i)$.

(1) Since the SkyCover set $S'(\Omega_\epsilon)$ is a subset of the set of representative points, $P(\Omega)$, i.e., $S'(\Omega_\epsilon) \subseteq P(\Omega)$, it is sufficient to show that $P(\Omega) \subseteq D(\Omega_\epsilon)$. Consider any point $t \in P(\Omega)$ lying in the grid cell $z = \langle z_0, \ldots, z_{d-1} \rangle$. Assuming non-uniform grid partitioning, the index structure ensures that $\forall i,\ x_i \leq z_i \leq y_i$, where $x_i = \lfloor \log_{(1+\epsilon)} l_i \rfloor$ and $y_i = \lfloor \log_{(1+\epsilon)} u_i \rfloor$. This implies that $\forall i,\ (1+\epsilon)^{x_i} \leq t_i \leq (1+\epsilon)^{y_i+1}$. Plugging in the values of x_i and y_i, and using the fact that for any real number a, $a - 1 \leq \lfloor a \rfloor < a$, we get the following: $\forall i,\ l_i/(1+\epsilon) \leq t_i \leq u_i.(1+\epsilon)$. Hence, $t \in D(\Omega_\epsilon)$. Therefore, $P(\Omega) \subseteq D(\Omega_\epsilon)$.

(2) Consider a range-constrained skyline point $s \in S(\Omega)$. Suppose it lies in a grid cell z. Assume the point t to be the cell representative of z where t may or may not be equal to s. Since s and t are in the same cell, the construction of the non-uniform grid partitioning ensures that t ϵ-covers s. More formally, $\forall i, t_i \leq (1+\epsilon)s_i$. If $t \in S'(\Omega_\epsilon)$, then s is covered by t. If, however, $t \notin S'(\Omega_\epsilon)$, then there must exist a point $u \in S'(\Omega_\epsilon)$ that dominates t, i.e., $u \succ t$. Combining with the above inequality, we get $\forall i, u_i \leq t_i \leq (1+\epsilon)s_i$, i.e., u ϵ-covers s. □

The ramification of this theorem is that even if $s \in S(\Omega)$ was missed, there is another point $t \in S'(\Omega_\epsilon)$ that approximates it in the sense that it is not too bad in any of the dimensions. To be precise, the values of t are within a multiplicative factor of $(1+\epsilon)$ from those of s in *every* dimension.

Figure 2 shows an example of how Theorem 1 works. Assume that $\Omega = [1, R)^2$ where $R = (1+\epsilon)^3$. In this case, any range-constrained skyline in $S(\Omega)$ is a skyline of D. The skyline point u, which is reported, ϵ-covers itself. The skyline t is missed and is represented by s which has a lower entropy (the equi-entropy line is shown as dashed). The reported point $u \in S'$ dominates s, and consequently, ϵ-covers t. Similarly, even though w is not reported, it is ϵ-covered by its cell representative $z \in S'$.

4.2 Falsely Reported Skylines

Even though the set $S'(\Omega_\epsilon)$ returned by the SkyCover framework correctly ϵ-covers the range-constrained skyline set $S(\Omega)$, it may happen that *not* every point returned is a range-constrained skyline itself, i.e., there may exist a point $u \in S'(\Omega_\epsilon)$ such that $u \notin S(\Omega)$. Figure 2 shows such a situation. The point v is not an actual skyline as it is dominated by t. It is also a grid representative, i.e., $v \in$

$P(\Omega)$. However, since $t \notin P(\Omega)$, and there is no other point that dominates v, $v \in S'(\Omega_\epsilon)$.

Unfortunately, the guarantees for the values of such falsely reported range-constrained skylines are not very strict. Along some of the dimensions, the values can be arbitrarily bad as compared to an actual range-constrained skyline point. However, the next theorem shows that it can still be guaranteed that the values of such falsely reported range-constrained skylines *cannot* be bad in *all* the dimensions.

THEOREM 2. *For any $u \in S'(\Omega_\epsilon)$, there does not exist any $s \in S(\Omega)$ such that $\forall i,\ u_i > (1+\epsilon)s_i$.*

PROOF. We prove by contradiction. Suppose such a point $s \in S(\Omega)$ exists, such that $\forall i,\ u_i > (1+\epsilon)s_i$. From Theorem 1, there must exist $t \in S'(\Omega_\epsilon)$ that ϵ-covers s, i.e., $\forall i,\ t_i \leq (1+\epsilon)s_i$. Together, this implies that, $\forall i, t_i < u_i$, i.e., $t \succ u$. Since $t \in S'(\Omega_\epsilon)$, therefore, $u \notin S'(\Omega_\epsilon)$, which is a contradiction. □

In Figure 2, even though the value of v is very large in the x-dimension as compared to the skyline point u, it is not worse than a ratio of $(1+\epsilon)$ in the y-dimension as well.

For real-life applications such as online hotel deals, assuming a value of $\epsilon = 0.05$, this implies that if there is an actual skyline hotel with cost 100 units and distance 2 units, no hotel is reported in the SkyCover set that has both cost more than 105 units and distance more than 2.1 units.

4.3 Expected Number of Representative Points

In this section, we analyze the expected number of representative grid points for the SkyCover framework. For the sake of comparison, we do the analysis for both uniform and non-uniform grids.

We assume that the points are generated independently and the values are independently and uniformly distributed along the dimensions across the data space.

There are two types of grid cells, empty and non-empty. Since no data point hashes to an empty grid cell, there is no representative for such cells. For non-empty cells, however, even if there are more points, exactly one point is chosen as the representative. Thus, we essentially need to calculate the number of non-empty grid cells to get an estimate of m.

4.3.1 Uniform Grids

For uniform grids, there are a total of g^d grid cells having the *same* volume. Hence, the probability that a point lies in a particular grid cell is

$$P_{point} = g^{-d}. \tag{6}$$

The probability that the cell remains empty is equivalent to the probability that none of the n points lie in it. Since the points are generated independently, this is equal to

$$P_{empty} = (1 - P_{point})^n = (1 - g^{-d})^n. \tag{7}$$

Thus, the expected number of empty cells is

$$\mathbf{E}[\text{empty}] = \sum_{\forall \text{cells}} P_{empty} = g^d(1 - g^{-d})^n. \tag{8}$$

The estimate for the total number of non-empty grid cells for uniform grid partitions is, therefore,

$$m_u = g^d - g^d(1 - g^{-d})^n. \tag{9}$$

Expressing Eq. (9) using binomial expansion, we get

$$m_u = g^d - g^d\left(1 - n.g^{-d} + \frac{n(n-1)}{2}.g^{-2d} - \dots\right)$$

$$\therefore\ \frac{m_u}{n} = 1 - \frac{n-1}{2}g^{-d} + \dots \tag{10}$$

When $n < g^d$, the later terms can be ignored, and therefore, with increasing n, the ratio m_u/n decreases. On the other hand, when $n \geq g^d$, since m is constrained to be at most g^d, the ratio will decrease when n increases. Thus, with increasing number of points and fixed number of grid cells, the proportion of representative points decreases.

When n is fixed, and g^d is increased, the proportion m_u/n increases as intuitively there are more options for a point to lie in, and consequently, more representative points are preserved. When $g^d \geq n$, the ratio will saturate to 1.

4.3.2 Non-Uniform Grids

The analysis for non-uniform grid partitions is not so straightforward as the grid cells have *different* volumes. Hence, the probability of a cell being empty depends on its *location*.

The volume of a grid cell $x = (x_0, \dots, x_{d-1})$ where $0 \leq x_i \leq g-1$ is the grid index along dimension i is

$$v(x) = \prod_{i=0}^{d-1}\left[(1+\epsilon)^{x_i+1} - (1+\epsilon)^{x_i}\right] = \epsilon^d(1+\epsilon)^{\sum_{i=0}^{d-1} x_i}$$
$$= \epsilon^d(1+\epsilon)^{\sigma_x} \tag{11}$$

where σ_x denotes the sum $\sum_{i=0}^{d-1} x_i$ for a cell x.

Since the total volume of the data space is $(R-1)^d$, the probability that a point lies in the grid cell x is $P_{point} = v(x)/(R-1)^d$. The probability that it remains empty when n points are generated independently is, therefore,

$$P_{empty} = (1 - P_{point})^n = \left(1 - \frac{v(x)}{(R-1)^d}\right)^n$$
$$= \left(1 - \left(\frac{\epsilon}{R-1}\right)^d(1+\epsilon)^{\sigma_x}\right)^n. \tag{12}$$

Thus, the expected number of empty cells is

$$\mathbf{E}[\text{empty}] = \sum_{\forall \text{cells}} P_{empty} = \sum_{\forall x}\left(1 - \tau^d(1+\epsilon)^{\sigma_x}\right)^n \tag{13}$$
$$= \sum_{\forall x}\left[\sum_{j=0}^{n}(-1)^j\binom{n}{j}\left(\tau^d(1+\epsilon)^{\sigma_x}\right)^j\right]$$
$$= \sum_{j=0}^{n}\left[(-1)^j\binom{n}{j}\tau^{jd}\sum_{\forall x}(1+\epsilon)^{j\sigma_x}\right]. \tag{14}$$

where τ denotes the ratio $\epsilon/(R-1)$.

To simplify the above equation, we denote $(1+\epsilon)^j$ as α. For the first term, i.e., when $j=0$, the sum $\sum_{\forall x}(1+\epsilon)^{j\sigma_x}$ is simply the total number of cells, which is g^d. For other terms, the summation can be computed by unrolling one dimension at a time. Thus,

$$\sum_{\forall x}\alpha^{\sigma_x} = \sum_{\forall x}\alpha^{\sum_{i=0}^{d-1} x_i}$$
$$= \sum_{\forall i=0,\dots,d-1;\forall x_i=0,\dots,g-1}\left(\alpha^{\sum_{i=0}^{d-1} x_i}\right)$$

Table 2: Values of β and m for different combinations.

g	d	g^d	ϵ	n	β	m_{est}
50	3	125000	0.01	10000	0.16	9588
20	4	160000	0.05	10000	0.34	9594
10	5	100000	0.05	10000	0.28	9471
7	6	117649	0.05	10000	0.19	9564

$$= \sum_{\forall i=1,\ldots,d-1;\forall x_i=0,\ldots,g-1} \left[\sum_{x_0=0,\ldots,g-1} \left(\alpha^{\sum_{i=1}^{d-1} x_i} . \alpha^{x_0} \right) \right]$$

$$= \sum_{\forall i=1,\ldots,d-1;\forall x_i=0,\ldots,g-1} \left[\left(\alpha^{\sum_{i=1}^{d-1} x_i} \right) . \sum_{x_0=0,\ldots,g-1} \alpha^{x_0} \right]$$

$$= \sum_{\forall i=1,\ldots,d-1;\forall x_i=0,\ldots,g-1} \left[\left(\alpha^{\sum_{i=1}^{d-1} x_i} \right) . \left(\frac{\alpha^g - 1}{\alpha - 1} \right) \right]$$

$$= \left(\frac{\alpha^g - 1}{\alpha - 1} \right) \sum_{\forall i=1,\ldots,d-1;\forall x_i=0,\ldots,g-1} \left(\alpha^{\sum_{i=1}^{d-1} x_i} \right)$$

$$= \cdots = \left(\frac{\alpha^g - 1}{\alpha - 1} \right)^d . \quad (15)$$

Using Eq. (15) in Eq. (14), and since $(1+\epsilon)^g = R$, we get

$$\mathbf{E}[\text{empty}] = g^d + \sum_{j=1}^{n} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{(1+\epsilon)^{jg} - 1}{(1+\epsilon)^j - 1} \right)^d$$

$$= g^d + \sum_{j=1}^{n} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{R^j - 1}{(1+\epsilon)^j - 1} \right)^d . \quad (16)$$

The computation of the exact value of the above expression is infeasible due to large n. However, if the expression $\beta_x = n(\tau^d(1+\epsilon)^{\sigma_x}) \ll 1$ for all cells x (from Eq. (12)), then the binomial series converges rapidly and can be bounded with very low error using only the first few terms. For example, the second term, (i.e., for $j = 1$) is simply n. Thus, a very crude estimate for the number of empty cells is $\mathbf{E}[\text{empty}] = g^d - n$.

Assume that $\beta = \max_{\forall x} \beta_x$ denotes the maximum value all among the grid cells, i.e., $\beta = n(\tau^d(1+\epsilon)^{\max_{\forall x} \sigma_x})$. When $\beta < 1$, the series can be cut-off before any positive term to get a lower bound. Thus, for example, by retaining only four terms (i.e., from $j = 0$ to 3), the estimate for the expected number of empty cells is at least

$$\mathbf{E}[\text{empty}] > g^d + \sum_{j=1}^{3} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{R^j - 1}{(1+\epsilon)^j - 1} \right)^d$$

$$= g^d - n + \binom{n}{2} \tau^{2d} \left(\frac{R^2 - 1}{(1+\epsilon)^2 - 1} \right)^d - \binom{n}{3} \tau^{3d} \left(\frac{R^3 - 1}{(1+\epsilon)^3 - 1} \right)^d . \quad (17)$$

This translates to an upper bound on the estimate of the expected number of representative points:

$$m_{nu} = g^d - \mathbf{E}[\text{empty}]$$

$$\lessapprox n - \sum_{j=2}^{3} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{R^j - 1}{(1+\epsilon)^j - 1} \right)^d . \quad (18)$$

where $\lessapprox$ signifies less than or equivalent.

Even if $\beta \not< 1$, the number of empty cells can be approximated by the first k terms of the binomial expansion:

$$\mathbf{E}[\text{empty}] \simeq g^d + \sum_{j=1}^{k} (-1)^j \binom{n}{j} \tau^{jd} \left(\frac{R^j - 1}{(1+\epsilon)^j - 1} \right)^d . \quad (19)$$

An appropriate estimate of m_{nu} can then be obtained.

Table 3: Estimates of m using Eq. (19) and their empirical counterparts; k denotes the number of terms after which m converges ($\beta = 1.5 \times 10^{-6} \times n$, $g = 8$, $d = 7$, $\epsilon = 0.05$, $g^d = 2097152$).

n	k	m_{est}	m_{emp}
1×10^4	3	9975	9977
5×10^4	4	49357	49354
1×10^5	5	97449	97463
5×10^5	7	440688	440592
1×10^6	10	782406	782116
5×10^6	23	1857515	1857442
1×10^7	40	2059755	2059738

Table 3 shows that even when $\beta > 1$, the series converges within a few iterations and the estimates thus obtained are quite close to the empirical ones.

Since the maximum σ_x for any cell is $(g-1)d$, the condition under which $\beta < 1$ is $n(\tau^d(1+\epsilon)^{(g-1)d}) < 1$. Note that evaluating Eq. (13) is computationally inefficient when the number of cells, i.e., g^d is too high. Hence, the approximation using Eq. (17) is needed for only such cases. Table 2 shows that $\beta < 1$ for typical values of d, g, ϵ and n when g^d is high. It also lists the corresponding estimates for m.

Similar to the case of uniform grids, when the number of grid cells is fixed, but the number of points n is increased, the ratio m_{nu}/n decreases. Again, this happens as more points now hash to the same cell. The ratio increases when number of grid cells is increased keeping n fixed.

The more interesting observation is when the error parameter ϵ is changed, keeping all the other parameters fixed. When ϵ increases, the ratio of the largest grid cell to the overall volume increases and that of the smallest decreases. In other words, the distribution of the volume becomes more skewed. Thus, more points are likely to lie on a smaller number of cells, thereby increasing the expected number of empty cells. Hence, the ratio m_{nu}/n decreases with increasing ϵ.

5. QUALITY METRICS

In this section, we describe the various metrics that can be used to assess the quality of an approximate or reduced skyline set.

The first metric, proposed in [32], is the maximum Jaccard similarity between any two sets of points dominated by a pair of skyline points. The lesser the Jaccard similarity, the more diverse the two skyline points are. For a particular size k, the more diverse the reduced skyline set is, the better.

The second measure, proposed in [23], is the ratio of points dominated by a subset of size k to the total number of points. The more this ratio is, the better.

5.1 Covering Factor

In this paper, we introduce two more quality metrics based on *covering factor*. Intutively, the covering factor is the ratio by which a skyline needs to be stretched in order to be dominated by a returned point in SkyCover. Hence. if $s \in S'$, $cf_s = 1$. Otherwise, it is the (maximum) approximation ratio over all dimensions, max-

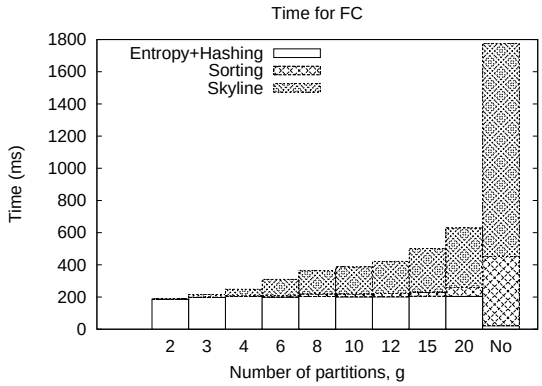


Figure 3: Running times for real data (FC).

imized over all the points $t \in S'$:

$$cf_s = \min_{t \in S'} \left\{ \max_i (t_i/s_i) \right\} \quad (20)$$

Thus, the covering factor essentially captures how well a skyline point $s \in S$ is covered by an approximate skyline point $t \in S'$.

5.2 Metrics Based on Covering Factor

Based on the covering factor, we propose two quality metrics:

1. *Worst Covering Factor (WCF)*, is the maximum covering factor of all the skylines.

2. *Cumulative Covering Factor (CCF)*, is the area under the curve of the cumulative distribution function (cdf) of the covering factors of all the skyline points.

Lower values of WCF indicate better quality of the solution. On the other hand, higher values of CCF signify that the covering factors attain their highest values earlier and are, thus, preferred.

6. EXPERIMENTAL RESULTS

In this section, we report in detail the results of experimenting with both real and synthetic datasets. All the experiments were done in Java on an Intel Core i7 CPU 870 @ 2.93 GHz machine with 8 GB RAM running Ubuntu Linux 12.04 LTS (64-bit).

For our empirical evaluation, we assume the worst case scenario of the query range Ω, i.e., $\Omega = [1, R)^d$. Hence, $D(\Omega) = D$, $S(\Omega) = S$, and $S'(\Omega_\epsilon) = S'$.

6.1 Real Dataset

The real dataset was the Forest Covertype (FC) dataset, containing different attributes of forest cover types, available from http://archive.ics.uci.edu/ml/datasets/Covertype. The size of the dataset is $n = 581012$. Similar to [32], only the first $d = 5$ attributes were considered. All the attributes were normalized to $[1, 2]$ space. The number of skylines is $s = 1356$.

6.1.1 Running Time

Figure 3 shows the running times when the number of grid partitions, g, is varied. It also includes the no-grid scheme. The total time is broken down into three components. The first is for grid partitioning (and entropy computation), the second is for sorting the m representatives, and the third is for the skyline computation routine. When g increases, m increases, thereby increasing the second and third components as well. As expected, the difference in skyline computation time is the main source of difference in the running time. When no grids are employed (i.e., the basic SFS method), the sorting and skyline finding times are too large. Overall, the SkyCover method runs 3-9 times faster.

6.1.2 Cardinality

Table 4 shows the statistics of the cardinality of skylines for the real dataset over the same range of g values. When m is low due to low g, a lot of actual skylines can be missed; however, a large proportion of them (the ratio w/s') always have an approximate skyline from the same grid cell. For high values of ϵ, k' approaches s. In this case, the number of skylines falsely reported ($v = k - k'$) is low as well.

6.1.3 Quality

The last set of experiments on real data measures the quality of the three methods, SkyDiver, k-RSP and our SkyCover, on the four quality metrics. Using a particular value of g, we first used our method to derive the SkyCover set. Assume that the cardinality is k. We then reduce the actual skyline set using SkyDiver and k-RSP to the same number k.

Figure 4 shows the covering factor metrics of the three methods. Expectedly, SkyCover is the best. Although the differences do not look significantly large, for the same k, while we ensure the WCF to be within $(1+\epsilon)$, SkyDiver and k-RSP routinely violate it. Thus, for these methods, no multiplicative error ratio can be guaranteed.

We also conducted a second set of experiments where we took a particular SkyCover set and reduced it to a subset of lower size, f, using the MH method of SkyDiver and the FM method of k-RSP. The skyline set is also reduced to the same size, f, using both SkyDiver and k-RSP. We then quantized all the four quality measures for varying f. We do it for $g = 8$ that produces $k = 333$.

Figure 5 shows the results. The MH method produces much better Jaccard similarity measures. However, very interestingly, when f is very small, using MH on SkyCover produces a better Jaccard measure than on SkyDiver. This shows that the SkyCover representation of the skyline set is extremely useful not only for the covering measures but also for other quality metrics. Similarly, FM produces better domination ratios and using FM on SkyCover is almost as good as k-RSP.

6.2 Synthetic Datasets

The synthetic datasets were generated using the code available from http://pgfoundry.org/projects/randdataset/. The parameters based on which the experiments were done are: (1) number of grid partitions per dimension, g, (2) error parameter, ϵ, (3) dataset cardinality, n, (4) dimensionality, d, and (5) type of dataset. While experimenting, we varied one parameter at a time while fixing the others to understand the effect of that single parameter better.

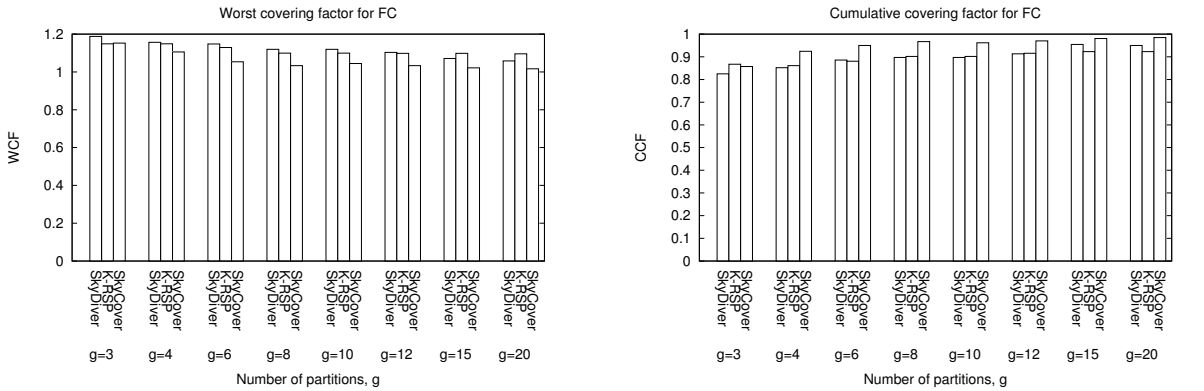
In the graphs, "NU" represents non-uniform grid partitioning, "Uni" represents uniform grid partitioning, and "No" represents the base method of computing the skylines without using grids.

6.2.1 Effect of Number of Grid Partitions

The first set of experiments (Figure 6a) measures the effect of number of grid partitions, g. For low values of g (till 5), the number of grid cells is so low ($g^d = 5^4 = 625$) that the ratio of the number of representative points m to the total number of points n is negligible. Consequently, the cardinality of the approximate skyline is very low as well and the entire method runs very fast. For medium values of g ($= 10$), when the number of grid cells ($g^d = 10^4$) is comparable to (but still less than) $n(= 50000)$, $m \to g^d$. In other words, almost all the grid cells are occupied. The running times are still lower than the base non-grid method. For high values of g

Table 4: Statistics of approximate skylines for the FC dataset: $n = 581012$, $s = |S| = 1356$, $d = 5$, $R = 2$.

Partitions per dimension	Error	Grid representatives	SkyCover size	Skylines truly reported	Skylines falsely reported	Skylines falsely missed	Covered by skyline in same cell	Covered otherwise
g	ϵ	$m = \|P\|$	$k = \|S'\|$	k'	$v = k - k'$	$s' = s - k'$	w	$t = s' - w$
2	0.414214	32	13	1	12	1355	1276	79
3	0.259921	172	30	11	19	1345	811	534
4	0.189207	515	87	45	42	1311	1102	209
6	0.122462	2086	173	108	65	1248	558	690
8	0.090508	5481	333	252	81	1104	651	453
10	0.071773	12014	427	285	142	1071	545	526
12	0.059463	22015	638	483	155	873	513	360
15	0.047294	44740	783	693	90	663	412	251
20	0.035265	103734	975	869	106	487	271	216



(a) Worst covering factor.

(b) Cumulative covering factor.

Figure 4: Variation of quality with number of partitions, g.

($= 25$), the number of grid cells is much larger ($\approx 4 \times 10^5$) compared to n and consequently, almost all the points lie in a separate grid cell by itself. As a result, $m \to n$. The running times for the grid-based mechanisms become worse due to the additional overhead of hashing, etc. There is little to choose between uniform and non-uniform grids as far as running time is concerned.

6.2.2 *Effect of Error Parameter*

For a fixed R, the error parameter ϵ is complementary to g. When g increases, ϵ decreases. Hence, the effect of ϵ is the reverse of g.

6.2.3 *Effect of Dataset Cardinality*

We next show how the algorithms scale with increasing size of dataset (Figure 6b). The cardinality is varied from 10^4 to 10^7 with $g^d = 8^7 \approx 2 \times 10^6$ grid cells. Interestingly, the ratio of skyline points to n, i.e., s/n decreases with increasing n (but not the absolute number). When n is small compared to g^d, $m/n \to 1$, and there is no gain either in running time or otherwise. (Note that both the scales in the figure are logarithmic.) When n approaches g^d (e.g., for $n = 5 \times 10^5$, i.e., when $n/g^d \approx 1/4$), the ratio m/n starts falling off, and an appreciable difference in the running time between using the grids and otherwise starts showing up. When n is larger than g^d, the ratio of m/n is quite low, and consequently, the grid-based mechanisms exhibit much better running times. At high n ($= 10^7$), the grid-based mechanisms are faster by a factor of more than 3. Also, since the hashing functions for the uniform case (which are division operations) are simpler than that for the non-uniform case (logarithms), the difference becomes significant at large n as there are $n.d$ such operations.

6.2.4 *Effect of Dimensionality*

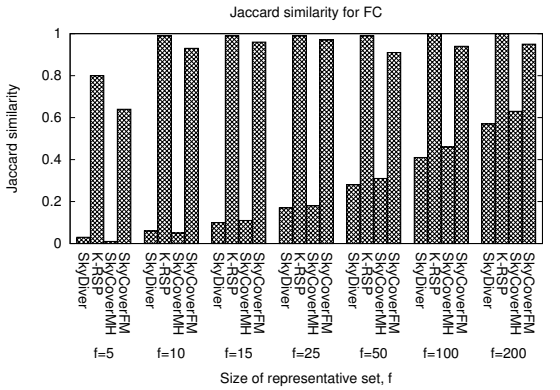
Figure 7a shows the effect of dimensionality, d. For independent datasets, the number of skylines grows exponentially with d. The running times follow the same behavior (note that the y-axis is logarithmic). At high d (from 10 onward), $m \to n$ due to too many grid cells ($g^d \sim 10^7$). Consequently, there is no gain in the running time by employing grids. For medium to low values of d (≤ 7), the running times are much better as both m and k are lower.

6.2.5 *Effect of Type of Data*

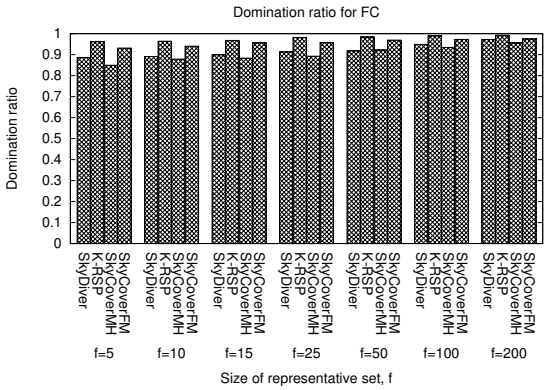
The last set of experiments is to gauge the effect of the type of the data. We generated three standard data types, independent, correlated and anti-correlated. For correlated data, the number of skylines is low as one point is likely to dominate many points. In an anti-correlated dataset, the skyline cardinality is high as it is unlikely that a point dominates another.

Interestingly enough, the number of representative grid cells, m, decreases for both correlated and anti-correlated datasets in relation to the independent one. The reason is that the data is not spread over the entire space uniformly, but is concentrated along certain directions. The correlated dataset is spread along the main diagonal; hence, many points fall on the largest grid $(g-1, \ldots, g-1)$ and m is lower. The anti-correlated dataset is spread along the main anti-diagonal and, hence, m is significantly larger than that of the correlated one but still lower than the independent distribution.

The running time for the anti-correlated dataset is much greater than independent though (Figure 7b). This is due to the fact that the high number of skylines makes the comparison step in SFS of an

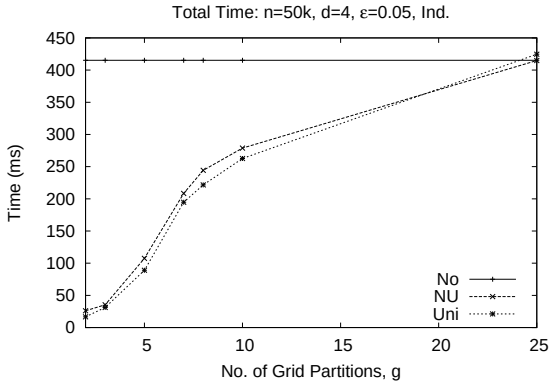


(a) Jaccard similarity.

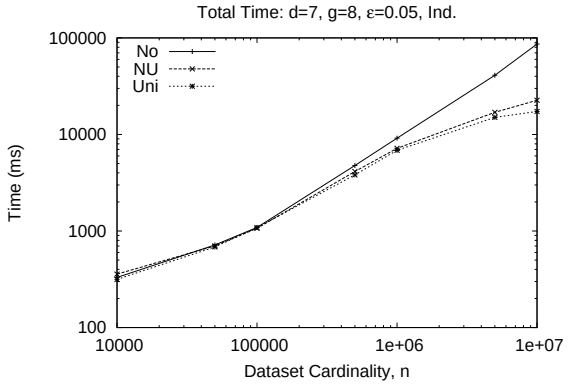


(b) Domination ratio.

Figure 5: Variation of quality with size of representative set, f.



(a) Number of partitions, g.



(b) Dataset cardinality, n.

Figure 6: Effect of g and n for synthetic data.

object against the current skylines much less efficient as the window size is larger. The gain in running times over the base method is the least for the independent dataset.

6.3 Summary of Experiments

We can summarize the empirical observations obtained from the experiments on synthetic data as follows. If all the other parameters are fixed, it is better to increase ϵ up to the factor that the application can tolerate. When g^d is much larger than n, then $m \rightarrow n$ and the grid-based mechanisms are not beneficial. On the other hand, if g^d is much less than n, then m is constrained by g^d and the number of approximate skylines retrieved is much lower than the actual skyline cardinality. The SkyCover guarantees of bounded multiplicative errors still holds, though. In general, the uniform grid-based partitioning method is faster than the non-uniform counterpart although it provides no guarantees. Finally, the number of grid representatives is the largest when the data dimensions are independently distributed, and our method works better for correlated and anti-correlated datasets.

7. CONCLUSIONS AND FUTURE WORK

Range-constrained skyline queries retrieve skyline points over a query range and is a generalization of the skyline query. In spite of having several applications, range-constrained skyline queries have not received much research attention. The reason is that even with a constrained range, the size of the skyline set can be impractically large. To address the above, in this paper, we introduced the concept of approximate range-constrained skyline queries, and proposed the SkyCover framework for efficiently computing them. The framework employs a hashing scheme based on non-uniform grid partitioning. The hashing itself guarantees the approximation bound on the desired skyline set, thus avoiding any separate approximate skyline computation technique. The hashing also significantly contributes towards efficiency.

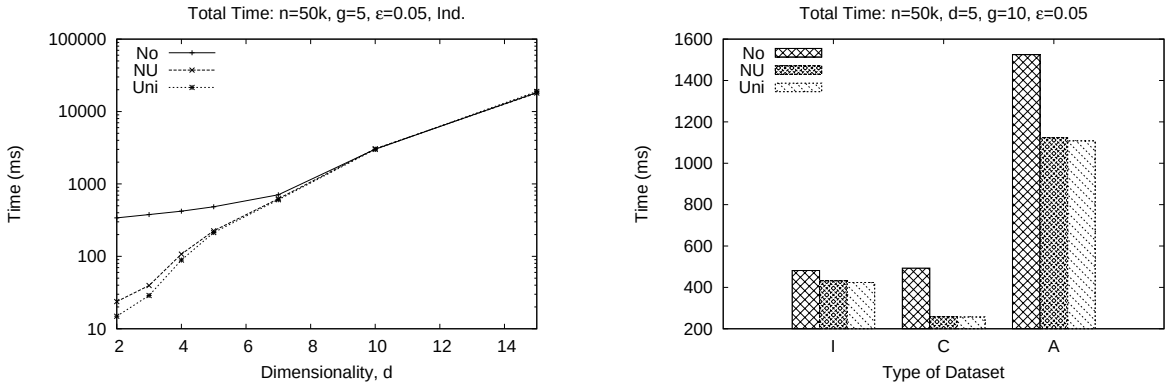
We also proposed two new metrics that can be used to measure the quality of an approximate skyline set. Empirical evaluation of our framework shows that it is significantly faster than the competing techniques, and yields solutions with high quality.

The grid-based hashing mechanism is generic enough to capture other kinds of errors including additive errors and no errors. More importantly, this framework can be applied in various settings such as data streams, and distributed environments. Detailed experimentation and analysis of such schemes remain a future work.

8. REFERENCES

[1] F. N. Afrati, P. Koutris, D. Suciu, and J. D. Ullman. Parallel skyline queries. In *ICDT*, pages 274–284, 2012.

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



(a) Dimensionality, d.

(b) Type of dataset.

Figure 7: Effect of d and type of dataset on synthetic data.

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

[4] C. Brando, M. Goncalves, and V. González. Evaluating top-k skyline queries over relational databases. In *DEXA*, pages 254–263. Springer, 2007.

[5] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *STOC*, pages 327–336, 1998.

[6] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *SIGMOD*, pages 503–514, 2006.

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

[8] A. Cosgaya-Lozano, A. Rau-Chaplin, and N. Zeh. Parallel computation of skyline queries. In *HPCS*, page 12, 2007.

[9] A. Das Sarma, A. Lall, D. Nanongkai, R. J. Lipton, and J. Xu. Representative skylines using threshold-based preference distributions. In *ICDE*, pages 387–398, 2011.

[10] E. Dellis, A. Vlachou, I. Vladimirskiy, B. Seeger, and Y. Theodoridis. Constrained subspace skyline computation. In *CIKM*, pages 415–424. ACM, 2006.

[11] H. Eder. On extending PostgreSQL with the skyline operator. Master's thesis, Vienna University of Technology, 2009.

[12] P. Flajolet and G. Nigel Martin. Probabilistic counting algorithms for database applications. *J. Computer and System Sciences*, 31(2):182–209, 1985.

[13] Y. Gao, J. Hu, G. Chen, and C. Chen. Finding the most desirable skyline objects. In *DASFAA*, pages 116–122, 2010.

[14] Z. Huang, Y. Xiang, and Z. Lin. l-SkyDiv query: Effectively improve the usefulness of skylines. *SCIENCE CHINA Information Sciences*, 53(9):1785–1799, 2010.

[15] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[16] W. Jin, J. Han, and M. Ester. Mining thick skylines over large databases. In *PKDD*, pages 255–266, 2004.

[17] A. K. Kalavagattu, A. S. Das, K. Kothapalli, and K. Srinathan. On finding skyline points for range queries in plane. In *CCCG*, 2011.

[18] V. Koltun and C. H. Papadimitriou. Approximately dominating representatives. *Theor. Comput. Sci.*, 371(3):148–154, 2007.

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

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

[21] H. Li, Q. Tan, and W.-C. Lee. Efficient progressive processing of skyline queries in peer-to-peer systems. In *InfoScale*, 2006.

[22] Z. Li, Z. Peng, J. Yan, and T. Li. Continuous dynamic skyline queries over data stream. *J. Computer Research and Development*, 1:014, 2011.

[23] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In *ICDE*, pages 86–95, 2007.

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

[25] C. Rhee, S. K. Dhall, and S. Lakshmivarahan. An optimal parallel algorithm for the maximal element problem (abstract). In *CSC*, page 435, 1990.

[26] J. B. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Nørvåg. Agids: A grid-based strategy for distributed skyline query processing. In *Int. Conf. Data Management in Grid and Peer-to-Peer Systems (Globe)*, pages 12–23, 2009.

[27] J. Selke and W.-T. Balke. Skymap: a trie-based index structure for high-performance skyline query processing. In *DEXA*, pages 350–365, 2011.

[28] L. Su, P. Zou, and Y. Jia. Adaptive mining the approximate skyline over data stream. In *ICCS*, pages 742–745, 2007.

[29] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *VLDB*, pages 301–310, 2001.

[30] Y. Tao, L. Ding, X. Lin, and J. Pei. Distance-based representative skyline. In *ICDE*, pages 892–903, 2009.

[31] Y. Tao, X. Xiao, and J. Pei. Efficient skyline and top-k retrieval in subspaces. *TKDE*, 19(8):1072–1088, 2007.

[32] G. Valkanas, A. N. Papadopoulos, and D. Gunopulos. Skydiver: a framework for skyline diversification. In *EDBT*, pages 406–417, 2013.

[33] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal, and A. El Abbadi. Parallelizing skyline queries for scalable distribution. In *EDBT*, pages 112–130, 2006.

[34] T. Xia, D. Zhang, and Y. Tao. On skylining with flexible dominance relation. In *ICDE*, pages 1397–1399, 2008.