Differentially Private K-Skyband Query Answering Through Adaptive Spatial Decomposition

Ling Chen, Ting Yu, Rada Chirkova

Differentially Private k-Skyband Query Answering through Adaptive Spatial Decomposition

Ling Chen[1], Ting Yu[2], Rada Chirkova[1]
[1]{lchen10, rychirko}@ncsu.edu,[2]tyu@qf.org.qa

[1] Department of Computer Science, North Carolina State University, Raleigh, USA
[2] Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar

Abstract. Given a set of multi-dimensional points, a k-skyband query retrieves those points dominated by no more than k other points. k-skyband queries are an important type of multi-criteria analysis with diverse applications in practice. In this paper, we investigate techniques to answer k-skyband queries with differential privacy. We first propose a general technique BBS-Priv, which accepts any differentially private spatial decomposition tree as input and leverages data synthesis to answer k-skyband queries privately. We then show that, though quite a few private spatial decomposition trees are proposed in the literature, they are mainly designed to answer spatial range queries. Directly integrating them with BBS-Priv would introduce too much noise to generate useful k-skyband results. To address this problem, we propose a novel spatial decomposition technique k-skyband tree specially optimized for k-skyband queries, which partitions data adaptively based on the parameter k. We further propose techniques to generate a k-skyband tree over spatial data that satisfies differential privacy, and combine BBS-Priv with the private k-skyband tree to answer k-skyband queries. We conduct extensive experiments based on two real-world datasets and three synthetic datasets that are commonly used for evaluating k-skyband queries. The results show that the proposed scheme significantly outperforms existing differentially private spatial decomposition schemes and achieves high utility when privacy budgets are properly allocated.

Keywords: k-Skyband Query · Differential Privacy · Adaptive Spatial Decomposition

1 Introduction

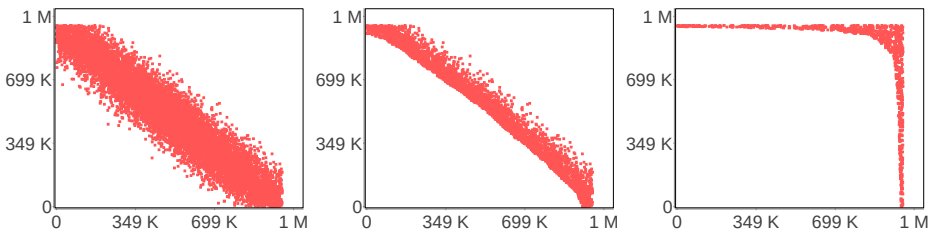
Given a set of multi-dimensional points, a k-skyband query [30] identifies the set of points that are *dominated* by at most k other points. A point p dominates another point q if p is at least as good as q on all dimensions and strictly better than q in at least one dimension. A k-skyband query is a generalization of a skyline query [5, 7, 25]: when k is 0, a k-skyband query is just a skyline query. As an important type of preference queries, skyband queries and their variants [15, 30] have wide applications in practice, e.g., location-based services [22] and service recommendations [23].

Similar to other data analysis tasks, directly releasing the results of k-skyband queries over sensitive data of individuals could result in privacy breach. For example, the presence (absence) of one point may cause a large set of points to be removed

from (included in) the k-skyband results. Thus, by analyzing the output of k-skyband queries, an adversary may infer the presence or absence of an individual in the dataset, which could be very sensitive. Due to such potential privacy risks, a data owner may be reluctant to share k-skyband query results with collaborators or the public, even if such sharing could bring significant benefits.

In this paper, we develop techniques to answer k-skyband queries with differential privacy [10, 11]. Unlike syntactic approaches such as k-anonymity [16, 31], differential privacy provides a provable strong privacy guarantee that the output of a computation is insensitive to any particular individual. That is, an adversary has limited ability to make inference about whether an individual is present or absent in the dataset.

We first propose a general technique BBS-Priv, which accepts any differentially private spatial decomposition tree as input and leverages data synthesis to generate private k-skyband results. Specifically, in a spatial decomposition tree, an internal node contains the coordinates of a region, the number of data points within the region (referred as point count), and pointers to its child nodes (i.e, subregions) at the lower level. Given a spatial decomposition tree, such as private quad-tree or kd-tree [9], BBS-Priv adopts the branch-and-bound paradigm to progressively traverse nodes for dominance checking, and prunes internal nodes that do not contain k-skyband points, i.e., there is no need to access all the partitions. When reaching a node that could not be further pruned, BBS-Priv generate approximate k-skyband results using synthesized points based on the node's point count.



(a) *Dataset* (b) *k-skyband* (c) *quadtree*

Fig. 1: Comparison of true and private k-skyband results with anti-correlated synthetic dataset when $k = 200$.

As several techniques have been proposed in the literature to generate private spatial decomposition trees [9], it seems that we can directly combine them with BBS-Priv to answer k-skyband queries privately. Unfortunately, such a straightforward approach would significantly distort k-skyband query results. Figure 1a shows an example synthetic dataset following an anti-correlated distribution, which is commonly used in skyline query evaluation. Figure 1b shows the true k-skyband results, and Figure 1c shows the private k-skyband results when combining BBS-Priv with a differentially private quadtree [9]. We see that private quadtree fails to sufficiently capture the properties of the dataset that are important to k-skyband queries, producing k-skyband results significantly different from the true results. There are two major reasons for such a poor performance. First, in k-skyband queries, the regions close to the upper-right corner are much more important than those lower-left regions, since these regions contain points with preferred values in all dimensions. Thus, it is much more desirable to accurately

capture data distributions in upper-right regions than the lower-left ones. Existing spatial decomposition schemes are designed for spatial range queries and thus all regions are treated equally, and thus are not suitable to answer k-skyband queries. Second, existing schemes achieve differential privacy by perturbing the point count in each region. After such perturbation, some empty regions' point counts may become positive. If these regions are close to the upper-right corner, these noisy points would distort the k-skyband results significantly.

Based on the above observations, we develop k-skyband tree, a novel spatial decomposition technique, which partitions data space adaptively based on the parameter k. The insight of k-skyband tree is that not all the regions contribute equally to the k-skyband results (e.g., points in dominance regions do not contribute to the k-skyband results at all), and thus finer and more accurate decompositions should be performed on the regions that are likely to contain k-skyband results. Built on this insight, when choosing a splitting point to partition a region, k-skyband tree finds an appropriate upper-right region ne that contains more than k points, which guarantees that the points in the lower-left region sw can be safely pruned. The upper-right region ne gets finer decompositions in subsequent splittings. We further present a suite of techniques to publish a k-skyband tree privacy, and propose a post-processing technique to improve its accuracy by suppressing data synthesis in those empty partitions whose noisy point counts become positive. We can then feed BBS-Priv with the private k-skyband tree to compute private k-skyband results.

For evaluation, we conduct experiments over three synthetic datasets with different distributions and two real-world datasets [1, 2], and compare private k-skyband trees with private quad-tree and private kd-tree, two well-known differentially private spatial decomposition schemes. Our results show that for synthetic datasets k-skyband tree outperforms quadtree and kd-tree when sufficient privacy budgets are allocated ($\epsilon > 0.5$). Further, our proposed technique significantly outperforms private quadtree and kd-tree in the two real datasets for all studied privacy budgets (ϵ ranging from 0.1 to 2.0). One key observation from our experiments is that, though the three synthetic datasets are commonly used in skyline query evaluation, they unfortunately do not capture the actual data distributions in real applications where k-skyband queries matter most. We observe that real datasets tend to have fan-shaped data distributions, where dominant points spread sparsely in dominance regions (e.g., the ne region in a 2-D space), while other inferior points more densely spread over other regions. Our scheme is adaptive enough to capture such distributions while existing spatial decomposition schemes fail to do so.

2 Preliminaries

2.1 Differential Privacy

Differential privacy [10] is a formal privacy model that guarantees the output of a query function to be insensitive to any particular record in the data set.

Definition 1. ***(ϵ-differential privacy)****: Given any pair of neighboring databases D and D' that differ in at most one individual record, a randomized algorithm A is ϵ-differentially*

private iff for any $S \subseteq Range(A)$*:*

$$Pr[A(D) \in S] \leq Pr[A(D') \in S] * e^{\epsilon}$$

The parameter ϵ is often referred as the privacy budget in differential privacy, as it directly affects the level of privacy protection. Obviously, the smaller ϵ, the harder to distinguish between D and D' from the output of A, and thus the stronger the privacy protection.

The most common strategy to achieve ϵ-differential privacy is to add noise to the output of a function. The magnitude of the noise is calibrated by the privacy budget ϵ and the sensitivity of the query function f, which is defined as the maximum difference between the outputs of the query function f on any pair of neighboring databases:

$$\Delta f = \max_{D,D'} \parallel f(D) - f(D') \parallel_1$$

There are two common approaches for achieving ϵ-differential privacy: Laplace mechanism [12] and Exponential mechanism [26].

Laplace Mechanism: The output of a query function f is perturbed by adding noise from the Laplace distribution with probability density function $pdf(x|b) = \frac{1}{2b}\exp(-\frac{|x|}{b})$, $b = \frac{\Delta f}{\epsilon}$. The following randomized mechanism A_l satisfies ϵ-differential privacy:

$$\mathcal{A}_l(D) = f(D) + Lap(\frac{\Delta f}{\epsilon})$$

Exponential Mechanism: This mechanism returns an output that is close to the optimum, with respect to a quality function. A quality function $q(D, r)$ assigns a score to all possible outputs $r \in R = range(f)$, and outputs closer to the true output receive higher scores. A randomized mechanism A_e that outputs $r \in R$ with probability

$$Pr[\mathcal{A}_e(D) = r] \propto exp(\frac{\epsilon q(D, r)}{2\Delta q})$$

satisfies ϵ-differential privacy, where Δq is the sensitivity of the quality function.

Differential privacy has two properties: *sequential composition* and *parallel composition*. Sequential composition is that given n independent randomized mechanisms $A_1, A_2, \ldots, A_n$ where A_i $(1 \leq i \leq n)$ satisfies ϵ_i-differential privacy, a sequence of A_i over the dataset D satisfies ϵ-differential privacy, where $\epsilon = \sum_1^n \epsilon_i$. Parallel composition is that given n independent randomized mechanisms $A_1, A_2, \ldots, A_n$ where A_i $(1 \leq i \leq n)$ satisfies ϵ-differential privacy, a sequence of A_i over a set of *disjoint data sets* D_i satisfies ϵ-differential privacy.

2.2 K-skyband Queries

Given a d-dimensional data set D, a k-skyband query returns all the points in D that are dominated by at most k other points in D. The dominance relationship in k-skyband queries is defined as follows:

Definition 2 (Dominance). *Given two d-dimensional points* $p = (u_1, \ldots, u_d)$ *and* $q = (v_1, \ldots, v_d)$*, if for all* $i = 1, \ldots, d$*,* $u_i \succeq v_i$ *and* $\exists j$*,* $u_j \succ v_j$*, we say that p dominates q (*$p \succ q$*), where* $\succ$ *denotes better than and* $\succeq$ *denotes better than or equal to.*

In k-skyband queries, k represents the thickness of the skyband results, and a skyline query [5, 30] is a special case of k-skyband queries when $k = 0$. k-skyband queries can be answered by extending algorithms for skyline queries, such as Branch-and-Bound Skyline (BBS) [30]. BBS is an efficient algorithm built on top of any spatial decomposition. A spatial decomposition is a hierarchical (tree) decomposition of a geometric space into smaller regions. In a spatial decomposition tree, an internal node stores the coordinates of a region, the number of points in that region (referred to as point count) and pointers to its child nodes, while the leaf nodes are individual data points.

Given a spatial decomposition tree, BBS traverses the nodes for dominance checking, and prunes internal nodes that are determined to contain no skyline points, i.e., not all the points will be accessed. It can be easily adapted to answer k-skyband queries by excluding a region if it is already dominated by k other points. In this paper, we focus on how to adapt BBS to answer k-skyband queries with differential privacy.

3 Approaches

3.1 BBS-Priv

BBS-Priv is inspired by the BBS algorithm [30]. BBS maintains a set S to keep track of all the k-skyband points discovered so far during the algorithm, and accesses the nodes in a spatial decomposition tree starting from the root node that covers the whole region. When a node n is accessed, if BBS finds more than k points in S that dominate n, n is pruned. Otherwise, BBS (1) inserts n into S if n is a leaf node that represents a point, or (2) expands n by accessing n's child nodes if n is an internal node. For n's child nodes, distances are computed according to L_1 norm, i.e., the *maxdist* of a point is the sum of its coordinates and the *maxdist* of an internal node is the *maxdist* of its upper-right corner point. These child nodes are inserted into a max heap that sorts nodes based on their *maxdist*, so that nodes with higher *maxdist* are accessed earlier. The intuition is that nodes with larger *maxdist* cannot be dominated by those with smaller *maxdist*, and thus BBS only needs to check the dominance relationship between the current accessed node and each point in S. The access order based on the max heap guarantees that points inserted into S are k-skyband points. BBS continues to access nodes from the max heap until the heap is empty, and returns points in S as the k-skyband results.

In BBS-Priv, the input is a differentially private spatial decomposition tree, whose leaf nodes are not individual points but a region. We can simply adapt BBS such that when reaching a leaf node e, if e is not pruned by points in S, we uniformly generate points in the region according to e's point count, treat each of them as a child of e, and continue. Due to space limit, we omit the detailed pseudocode of BBS-Priv.

Privacy Analysis. It is easy to see that BBS-Priv only conducts post processing of a spatial decomposition T. As long as T is constructed with differential privacy, BBS-Priv would provide the same privacy guarantee.

3.2 Differentially Private K-Skyband Tree

Although BBS-Priv can be combined with any existing differentially private spatial decomposition trees, as we will show in Section 4 later, the resulting k-skyband results

are often highly distorted compared with the true results. The reason is that existing differentially private spatial decomposition schemes aim to capture the distribution of all the data. For example, in existing schemes, it is common that a dense region (with high point count) will be further partitioned. However, for k-skyband queries, we may not need to do so if that area is already dominated by more than k points. Similarly, for a sparse region, if it is close to the upper-right corner, we may still need to continue the partition as it is likely to contain k-skyband results and we need to better capture their distributions. Based on this observation, we propose a novel spatial decomposition algorithm k-skyband tree specifically tailored to answer k-skyband queries, and will show further how to build k-skyband tree with differential privacy. For simplicity and easy explanation, we present our scheme for handling spatial data (2-D data). It can be easily extended to handle multi-dimensional data, which we omit due to space limit.

K-Skyband Tree The insight of k-skyband tree is to perform finer and more accurate decompositions on the regions that are more important for k-skyband results, i.e., the regions close to the upper-right corner. Note that when we say "upper-right corner", it is relative to the input dataset instead of to the whole space. Therefore, the partition of space must be dependent on the dataset. Specifically, k-skyband tree chooses a splitting point s such that there are more than k data points in the the upper-right region ne and all the data points falling into the dominance region sw can be excluded from the computation of k-skyband results. Details of k-skyband tree is presented at Appendix A. The main point of k-skyband tree is not to make query answering more efficient. Instead, it would guide fine-grained decomposition towards those regions that are likely to contain k-skyband results, so that when we add noise later to satisfy differential privacy, the distribution of points in those regions is preserved better, reducing the distortion of the true results.

Differentially Private K-Skyband Tree To make the process of building k-skyband tree satisfy differential privacy, we need to revise the algorithm in the following major steps. First, as the tree will reveal the split points for each region (i.e., the coordinates of internal nodes), we need to make the splitting process differentially private. Specifically, we leverage the Exponential Mechanism to choose private values for the splitting point. For s_x (s_y), we divide the possible output range, $[x_{min}, x_{max}]([y_{min}, y_{max}])$, into intervals based on the ranks of the true data points, and assign higher probability to the intervals closer to s_x (s_y). Once an interval is chosen based on the Exponential Mechanism, a value is uniformly sampled from the interval to be the private value of the splitting point.

Second, the point count of each node in the tree will reveal the number of points in each region. To achieve differential privacy, we adopt the laplace mechanism to add noise. Specifically, we add Laplacian noise to the point count of each node, protecting the true count of the points falling into the split regions.

The pseudo code of building differentially private k-skyband tree is shown in Algorithm 1 and the function *splitByK* is shown in Algorithm 3 (in Appendix A). Besides the region r, k in k-skyband queries, and the max height h, the algoritm accepts as input the privacy budgets $\epsilon_0, \ldots, \epsilon_h$ for each level of the spatial decomposition tree and the split

rate α used for computing the budget of choosing splitting points. In Algorithm 3, we apply the Exponential Mechanism to obtain noisy values for s_x and s_y (Lines 14-15), and use the obtained random values to form the noisy splitting point. In Algorithm 1, Line 1 and Lines 17-18 correspond to adding Laplacian noise to point counts.

Algorithm 1 Differentially Private k-skyband tree

Input: A region r, k in k-skyband queries, max height of the tree h, privacy budgets for each level of the tree $\epsilon_0, \ldots, \epsilon_h$, split rate α
Output: A differentially private spatial decomposition tree T

1: $r.ncount$ = $r.count$ + Lap($\frac{1}{\epsilon_0 * (1-\alpha)}$)
2: Q.enqueue(r)
3: **while** Q is not empty **do**
4: n = Q.dequeue()
5: **if** isLeaf(n, h) **then**
6: updateNoisyCount(n)
7: **continue** // back to Line 3
8: l = $n.level$, ϵ_c = $\epsilon_{l+1} * (1-\alpha)$, k' = $k + 1 + \frac{\sqrt{2}}{\epsilon_c}$
9: **if** $n.ncount > k'$ and $n.parent.midPointSplit$ is $false$ **then**
10: N = splitByK($n, k', \epsilon_l * \alpha$)
11: **if** $N.ne$ = n **then**
12: N = splitByMidPoint(n)
13: $n.midPointSplit$ = $true$, ϵ_c = ϵ_{l+1}
14: **else**
15: N = splitByMidPoint(n)
16: $n.midPointSplit$ = $true$, ϵ_c = ϵ_{l+1}
17: **for** $c \in N$ **do**
18: $c.ncount$ = $c.count$ + Lap($\frac{1}{\epsilon_c}$)
19: Q.enqueue($N.ne, N.nw, N.se$)
20: **if** $N.ne.ncount \leq k$ **then**
21: Q.enqueue($N.sw$)
22: **return** N

The algorithm starts by adding Laplacian noise to the point count of the input region r, and adds the root node into the queue Q for splitting (Lines 1-2). The algorithm splits the node recursively until there are no nodes left in Q. For each node n to be split (Line 4), the algorithm first computes the budget ϵ_c for obtaining noisy count and k' based on k and ϵ_c (Line 8). The reason why we use $k' = k + 1 + \frac{\sqrt{2}}{\epsilon_c}$ instead of $k + 1$ is because (1) by adding Laplacian noise with mean 0 to k, there is 50 % of the chance that we would obtain a noisy value smaller than $k + 1$; (2) if the noisy count of ne is smaller than $k + 1$, then sw cannot be pruned and we need to further split sw; (3) the standard deviation of Laplacian noise based on ϵ_c is $\frac{\sqrt{2}}{\epsilon_c}$ and adding this standard deviation to k to obtain k' (i.e., making the count of ne to be slightly larger than k) can make the noisy count of ne more likely larger than k; Based on k', the algorithm splits the node n using the corresponding budget in the level of the node n (Line 10). After splitting, for each split node, the algorithm computes the noisy count based on the count budget (Lines 17-18). If the node is split using midpoint as quadtree, the splitting budget is saved and the count budget is updated (Lines 11-13 and 15-16), and all the children nodes are set

to use midpoint split (Lines 13 and 16). If the noisy count of ne is less than or equal to k, the dominance region sw needs to be further split (Lines 20-21).

When a node's level reaches the maximum height of the tree (h), the node is considered as a leaf node and no further split is applied (Line 5). Also, if the noisy count of the node is too small (e.g., less than 8), further splitting the node will caused the counts of child nodes to be distorted significantly by the Laplacian noise, and thus the algorithm considers the node as a leaf node. In this case, if the leaf node is not at the maximum level, the remaining budgets allocated for the rest of the levels are used to recompute the noisy count [9] (Line 6). The algorithm terminates when there are no nodes left in Q.

Obtaining Splitting Point with Differential Privacy. To protect the values of the splitting point, the algorithm leverages the Exponential Mechanism (EM) [26] to sample the noisy values.

Let $L = \{x_1, \ldots, x_m\}$ be a set of m values in ascending order in some domain range $[lo, hi]$, and let x_s be the desired value of the splitting point. Let $rank(x)$ denote the rank of x in L, representing the number of items in L that are smaller than x. The quality function fed into the EM [9] is:

$$q(L, x) = -|rank(x) - rank(x_s)|,$$

The EM returns x with $Pr[EM(L) = x] \propto e^{-\frac{\epsilon}{2}|rank(x)-rank(x_s)|}$ based on this quality function. Since all values x between two consecutive values in L have the same rank, they are equally likely to be chosen, which can be implemented using uniformly random sampling between the two consecutive values in L. Accordingly, EM can be implemented by choosing an output from the interval $I_k = [x_k, x_{k+1})$ with probability proportional to $|I_k|e^{-\frac{\epsilon}{2}|k-rank(x_s)|}$. Once an interval I_k is chosen, EM then returns a uniformly random value in I_k.

Privacy Analysis Due to space limit, we provide next only a sketch of the proof of the privacy guarantee of private k-skyband tree. Note that, the construction process of k-skyband tree tree falls into the category of hybrid spatial decomposition as defined in [9], where in the first few levels it uses data-dependent split (i.e., SplitByK (Algorithm 3)) and in the remaining levels it uses data-independent split (i.e., through midpoints). Therefore the analysis of k-skyband tree's privacy guarantee is very similar to that in [9].

To construct a k-skyband tree with differential privacy, we need to combine the privacy guarantees of both tree structures and point counts. Note that adding or deleting a single data point changes the counts of all the nodes on the path from the root to the leaf containing that data point, and it could also affect the node splitting in the levels where data-dependent split is used. For a k-skyband tree with max height h, our algorithm assigns the privacy budget $\epsilon_0, \ldots, \epsilon_h$ for the nodes at level $0, \ldots, h$. To protect both node splitting results and point counts, for nodes at the level i ($0 \leq i \leq h$), our algorithms uses the Exponential Mechanism with budget $\alpha\epsilon_i$ to obtain noisy splitting points and adds Laplacian noise with budget $(1 - \alpha)\epsilon_i$ to obtain noisy point counts. For the levels where data-independent split is used, there is no need to protect node splitting, and the budget for splitting is also used for computing noisy counts. If

nodes n_1 and n_2 are not on the same root-to-leaf path, their point counts are independent of each other, and knowing the noisy counts of n_1 does not affect the privacy guarantees of n_2. Thus, based on the parallel composition property (Section 2.1), k-skyband tree at the level i satisfies ϵ_i-differential privacy. Further, based on the sequence composition property of differential privacy in Section 2.1, for $\epsilon = \epsilon_0 + \ldots + \epsilon_h$, the whole k-skyband tree satisfies ϵ-differential privacy.

Budget Allocation Strategy For budget allocation, we follow the geometric scheme proposed in [9]. Let ϵ_i denote the budget for level i of the tree, and the budgets for each level is computed using $\epsilon_{i+1} = 2^{\frac{1}{3}}\epsilon_i$. The intuition is that for nodes at the levels closer to root, the point counts are larger and more resistant to noise, and more budgets should be allocated for levels close to leaves.

For a level i where data-dependent split is used, we allocate $\alpha\epsilon_i$ to compute the noisy splitting point and $(1-\alpha)\epsilon_i$ to obtain noisy point counts. Previous study [9] shows that a small portion of budget is enough for splitting, and we set $\alpha = 10\%$ based on our empirical evaluations. Such allocation is also consistent with the study on private kd-tree [9], which shows that finding a splitting point using Exponential Mechanism requires less budget than adding Laplacian noise to point counts.

Post Processing Post processing is commonly used in differential privacy to improve utility [10, 13]. For example, [9, 19] leverages post-processing to improve count query accuracy. However, the utility of k-skyband queries do not solely depend on the accuracy of point counts, and the existing post-processing techniques optimized for count queries do not work properly for k-skyband queries. The reason is that k-skyband tree uses data-dependent split at the first few levels, and certain dominance regions of the noisy splitting point (i.e., sw regions) are excluded in the computation of k-skyband results. If we adjust the noisy counts of the nodes using existing post-processing techniques, the noisy counts of some ne regions may become less than k, making the corresponding sw regions not qualified for pruning. Due to such inconsistency between the region splitting and noisy counts, the properties of the dataset that are important to k-skyband queries cannot be sufficiently captured, further distorting the k-skyband results significantly.

Empirically, we observed that the major factor affecting the utility of private k-skyband results is to synthesize data points for the regions whose noisy counts are positive but their true counts are zero. If the regions of these nodes are close to the upper-right point (x_{max}, y_{max}), then the private k-skyband results are significantly distorted since the data points in these regions could dominate the data points in any other region.

To smooth such errors caused by data synthesis with Laplacian noise, we propose a novel post-processing technique. The insight is to not synthesize points for half of the empty leaf nodes, as Laplacian noise has a 50% of probability to be positive (or negative) and could turn half of the empty leaf nodes to have positive noisy counts. However, the number of empty leaf nodes partially reveals the data distribution of the data set. Therefore, instead of directly using the number of empty leaf nodes, we use the number of leaf nodes whose noisy count are negative to approximate the number of empty leaf nodes whose noisy count are positive.

Let C be the set of nodes whose true point counts are 0. Denote C_p as the subset of nodes in C whose noisy point counts are positive, and denote C_n as all the leaf nodes (not only those in C) whose noisy point counts are negative. As Laplacian noise has a 50% probability to be positive, the expected size of C_p is $E[|C_p|] = \frac{|C|}{2}$. C_n includes two types of nodes: (1) the empty leaf nodes whose noisy counts are negative ($C_{n,1}$) and (2) the leaf nodes whose point counts are positive but noisy counts become negative ($C_{n,2}$). Ideally, we should use $|C_{n,1}|$ to approximate $|C_p|$. But $|C_{n,2}|$ is usually a small number since we do not split nodes whose noisy counts are too small, we can ignore $|C_{n,2}|$ and directly use $|C_n|$ to approximate $|C_p|$.

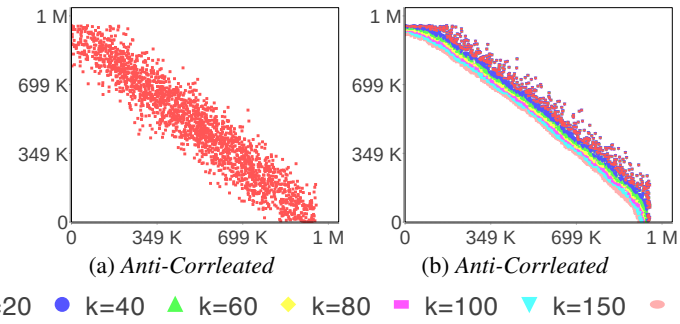
Based on the analysis, with the private k-skyband tree, we compute the number of leaf nodes whose noisy counts are negative ($|C_n|$), sort the leaf nodes with noisy positive counts in L_C in ascending order, and set the noisy count of the first $|C_n|$ nodes in L_C to zero. In this way, we can reduce the error of data synthesis in the nodes in C_p.

Another alternative approach is to directly compute $|C|$, and set the noisy count of the first $\frac{|C|}{2}$ nodes in L_C to zero. However, such approach relies on the true count of leaf nodes, which requires further privacy protection that consumes another portion of the privacy budget. Thus, we choose the approach based on $|C_n|$.

4 Evaluations

In our evaluations, we compare the performances of three techniques: k-skyband tree, kd-tree, and quadtree. For k-skyband tree, we limit the max tree height to be 7, and set the noisy count threshold to 8 to stop splitting. In this way, the max height allows k-skyband tree to get fine enough decompositions and the noisy count threshold prevents the noisy count of some nodes becoming too small and too sensitive to noise. For quadtree, we limit the max tree height to be 7, same as k-skyband tree. For kd-tree, we use the hybrid tree with the default parameters from the existing work [9], which was shown to perform the best for answering count queries. We conduct experiments with privacy budgets ranging from 0.1 to 1.0 and k ranging from 0 to 200; for each budget and each k, we apply the techniques on each dataset 10 times and report their average.

Datasets. The evaluations are carried out over three synthetic datasets that are commonly used for evaluating many interesting variations of skyline queries. They follow independent, correlated and anti-correlated distributions respectively [5]. Each of these datasets contains 10,000 points. For these synthetic datasets, we normalize the values to be in the range $[0, 1000000] \times [0, 1000000]$, and assume the larger values to be preferred in each dimension. Besides synthetic datasets, we also conduct experiments over two real-world datasets: NBA [2] and forest cover type [1]. The NBA dataset includes the statistics of all NBA players from 1997 to 2016, and there are 8645 points in total. In the NBA dataset, we want to find out NBA players who can score high points (in the range $[0, 3156]$) and get many rebounds (in the range [0, 1449]). The second dataset is uniformly sampled from the forest cover type dataset, which provides basic information for forested lands in the United States. It contains about 50,000 records and has been used to evaluate skyline query answering schemes [35]. From this dataset, we want to find out those forests located in uninhabited areas, i.e. those with high elevations (in the



(a) *Anti-Corrleated* (b) *Anti-Corrleated*

Fig. 2: Illustration of the synthetic dataset following anti-correlated distributions and their k-skyband results when k ranges from 20 to 200.

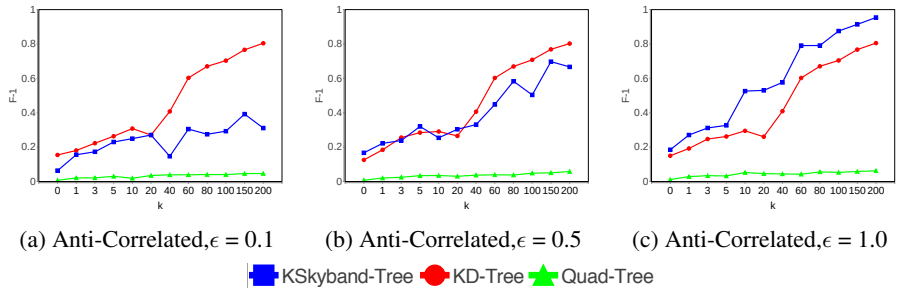
range $[1859, 3858]$) and long distance to roadways (in the range $[0, 7117]$), since those areas might exist some rare or endangered animal species for research.

The distributions of the synthetic datasets and the real datasets as well as their true skyband query results are shown in Figures 2 and 5. When k increases, the k-skyband expands toward the area containing less preferred points. Figures 2b and 5c-5d show different k-skybands on different datasets when k increases from 0 to 200. If $k_1 > k_2$, k_1-skyband results are the super set of k_2-skyband results, and visually a new stripe in a different color is added when k increases from k_2 to k_1.

Utility Metric. We use $F1$-measure to examine the similarity between the true k-skyband results S_t and the private k-skyband results S_p. To compute $F1$-measure, we first define false positives and false negatives based on the distance among points in S_p and S_t. Intuitively, with differential privacy, we could not guarantee that any true k-skyband results are returned. Instead, if a private skyband point is close to a real skyband point, then we say it is a hit (a true positive). Otherwise, it is a false positive. Similarly, if a true skyband point is not hit by any private skyband point, then it is counted as a false negative. More formally, given t_x and t_y, a point q in S_p is a *true positive (TP)* if there exists a point p in S_t such that $|p.x - q.x| \leq t_x$ and $|p.y - q.y| \leq t_y$; otherwise, we say q is a *false positive (FP)*. Similarly, a point p in S_t is a *false negative (FN)* if there exists no point q in S_p such that $|p.x - q.x| \leq t_x$ and $|q.y - p.y| \leq t_y$. Here for simplicity we use t_x and t_y instead of a radius to quantify the threshold of distance between a skyband point and its true positives. We refer to t_x (t_y) as the error tolerance threshold in x (y) dimension. Based on the counts of *TP*, *FP* and *FN*, we can compute the precision and recall and further derive $F1$-measure.

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN} \quad F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

For each of the five datasets, we obtain the ranges for each dimension (i.e., $[x_{min}, x_{max}]$ and $[y_{min}, y_{max}]$). We then set t_x and t_y to be 1%, 3%, 5%, and 7% of $(x_{max} - x_{min})$ and $(y_{max} - y_{min})$, and compute $F1$-measure accordingly. Note that for the 2-dimension dataset, when both t_x and t_y are set to 1% of the data domains, the error tolerance rate in the 2-dimension space becomes $1\% \times 1\%$, which requires a private point to be very close to a true point. In Sections 4.1 and 4.2, we show the $F1$-mesaure with both t_x and t_y set to 3%, and the results of different error tolerance rates are shown in Appendix B.



(a) Anti-Correlated,ϵ = 0.1 (b) Anti-Correlated,ϵ = 0.5 (c) Anti-Correlated,ϵ = 1.0

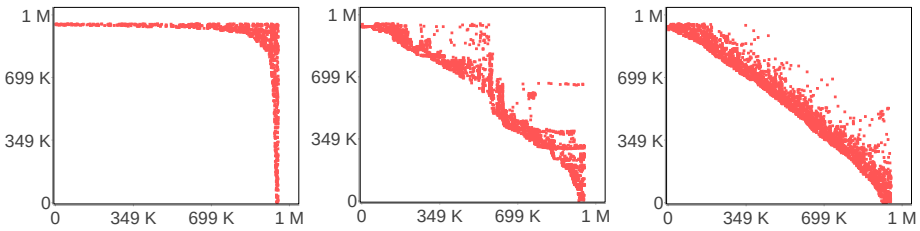
Fig. 3: Comparing $F1$-measure of k-skyband tree, kd-tree and quadtree on the anti-correlated dataset when k ranges from 0 to 200 and ϵ ranges from 0.1 to 1.0.

4.1 Results on Synthetic Datasets

Figure 3 shows the $F1$-measure results for the anti-correlated synthetic datasets. Due to space limit, we omit the results for the normal and correlated distributions. In the figure, the x-axis shows different values for k and the y-axis shows the values of $F1$-measure.

Impacts of datasets. For the anti-correlated dataset, quadtree has very poor $F1$-measure scores (about 0.1). Unlike the independent and correlated datasets, the true k-skyband points are concentrated in the center areas instead of the areas close to the upper-right point, as shown in Figure 2b. Thus, when quadtree fails to capture the properties of those regions that contribute most to k-skyband results, the resulting private k-skyband points are still in the regions close to the optimal point (as shown in Figure 4a), causing high false positives and high false negatives, leading to $F1$-measure scores. kd-tree performs better than k-skyband tree when ϵ is less than or equal to 0.5. For example, when ϵ is 0.1, k-skyband tree uses very little budget for splitting the region and cannot capture the data distribution precisely. The resulting k-skyband is similar to the one shown in Figure 4a. However, when ϵ becomes larger, which allows enough budget for splitting the region, k-skyband tree performs much better than the other two trees (shown in Figures 4a and 4b), and produces the private k-skyband (shown in Figure 4c) that is very similar to the true k-skyband. Though both k-skyband tree and kd-tree are data dependent and will adaptively conduct finer decompositions in dense regions, k-skyband tree focuses the decompositions on the regions that are most important to k-skyband queries (i.e., the upper-right regions), while kd-tree treats each region equally and does not provide fine enough decompositions in the upper-right regions, which explains the accuracy gap between kd-tree and k-skyband tree, especially when ϵ becomes bigger. For the independent and correlated datasets, the results are similar.

Impacts of k. Generally, $F1$-measure improves with the increase of k, and k-skyband tree is better than the other two approaches when budgets are ¿ 0.5. The reason is that when k is small (< 20), k-skyband query results intend to contain only a few data points. For privacy protection, a relatively large amount of noise is needed to hide the exact locations of these data points, causing high distortion of actual query results. When k increases, more and more data points are contained in k-skyband query results. k-skyband tree would more accurately capture the distribution of skyband points, even if their exact locations are protected, which explains the increased utility. In this



(a) *quadtree* (b) *kd-tree* (c) *k-skyband tree*

Fig. 4: Private k-skyband results of the anti-correlated dataset when $\epsilon = 2$ and $k = 200$.

case, producing an accurate k-skyband is very difficult because the corresponding nodes with small number of points are sensitive to the added noise; when k becomes larger, the number of points in k-skyband becomes larger, and the corresponding nodes with larger number of points are more resistant to the added noise.

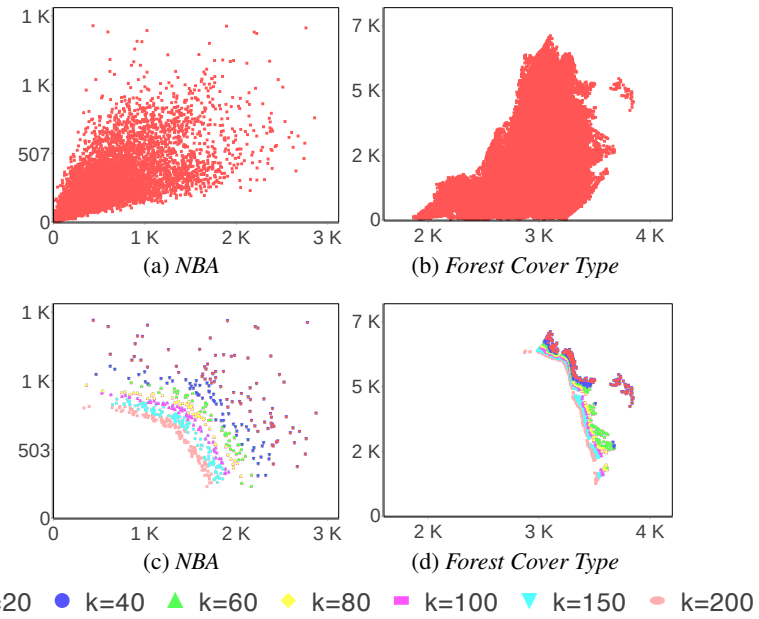
Summary. From the synthetic datasets, it does not seem k-skyband tree offers significant advantages over kd-tree or quadtree (depending on which datasets we look at). However, we note that these three datasets are widely used in past work to evaluate the *efficiency* and *scalability* of algorithms to compute exact skyline/k-skyband points. In this work however we focus on the accuracy of differentially privacy algorithms. We argue that none of the distributions in the three synthetic datasets are representative of practical datasets where k-skyband queries are meaningful and useful. For example, in the independent dataset, all the points evenly spread in the whole region, which means k-skyband points are as common as any other non-skyband points. Similarly, for the correlated dataset, it implies that if a point is superior in one dimension, it also tends to be so in the other. In that case, there would be no need to have k-skyband queries over multiple dimensions, as we only need to query points superior in one dimension and their superiority in the other dimension is implicitly ensured. The anti-correlated dataset goes to another extreme: if a point is superior at one dimension, it must be poor at the other, which also renders k-skyband queries over multi-dimensions unnecessary. Essentially k-skyband queries are to find *unusual* points who are good at both dimensions. Unusual points mean they cannot be as common as other points (inferior at both dimension) as in the independent and correlated datasets, and, on the other hand, they do exist (i.e., superior in both dimensions), not as in the anti-correlated dataset.

4.2 Results on Real Datasets

Figures 6a-6f show the $F1$-measure results for the two real datasets when varying k under different privacy budget ϵ.

The first thing we notice is that the distributions of the two real datasets (shown in Figure 5) do not resemble any of the three synthetic datasets (shown in Figure 2). Most of the points are "ordinary". They are not good at either dimension (i.e., they are largely concentrated around regions that are inferior in both dimensions, and k-skyband points instead spread out sparsely: we do have points that are superior in one dimension or in both dimensions, but they are not as concentrated as those "ordinary" points.

From the $F1$-measure results, for all ϵ values, we can see that k-skyband tree clearly outperforms the other two approaches when $k > 20$ in the NBA dataset; in the forest



(a) *NBA* (b) *Forest Cover Type*

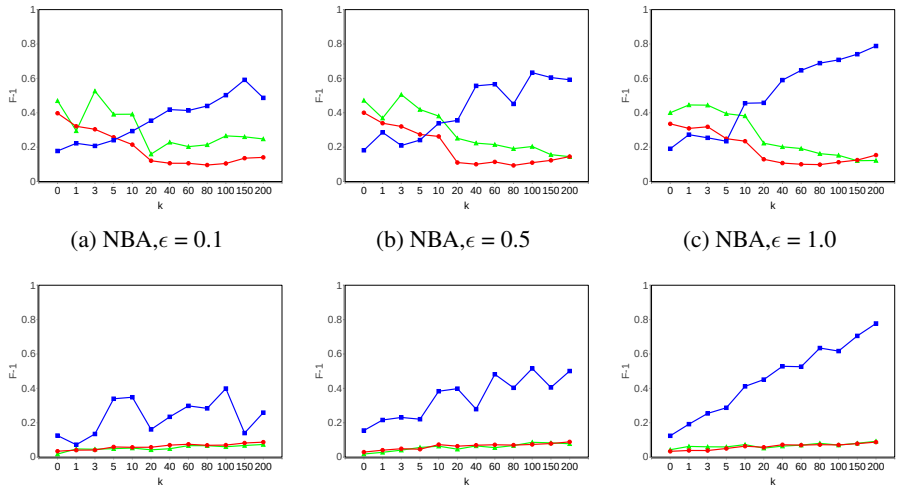
(c) *NBA* (d) *Forest Cover Type*

Fig. 5: Illustration of NBA and Forest Cover Type datasets and their k-skyband results when k ranges from 20 to 200.

cover type dataset, k-skyband tree achieves the best performance for all k values. The major reason is that both kd-tree and quadtree focus on splitting more in dense areas, which unfortunately in the real datasets correspond to regions that are not likely to contain k-skyband results. On the other hand, for the regions containing k-skyband points, since they are sparse, kd-tree and quadtree can only generate very course-grained partitions close to the optimal point or along each dimension. The consequence is that during data synthesizing phase, many points will be generated quite near the optimal point or the x and y axes. These synthesized points will be very likely to be included in the private k-skyband query results, which are far different from the real results. As a contrast, k-skyband tree will quickly prune out those dense but not interesting regions (from k-skyband queries' point of view) and split more in regions that likely contain k-skyband points even if these regions are not dense.

Summary. In general, k-skyband tree outperforms the other two trees for both real datasets for all ϵ values when k is reasonably large ($k > 20$). In the forest cover type dataset, k-skyband tree are better than the other two trees for all k values. Such results show that k-skyband tree achieves high utility not only in synthetic datasets used for evaluating various skyline computations, but also in real-world datasets used in multi-criteria decision making.

For these real-world datasets, we can observe that the desired data points usually are in the sparse areas that contain small number of points, and most of the data points are condensed in the areas that represent less desirable values. For example, in the NBA dataset, the best players spread in the areas that represent high scoring or high rebounding, and only a few are in the region that represents both high scoring and high



(a) NBA,$\epsilon = 0.1$ (b) NBA,$\epsilon = 0.5$ (c) NBA,$\epsilon = 1.0$

(d) Forrest Cover Type,$\epsilon = 0.1$ (e) Forrest Cover Type,$\epsilon = 0.5$ (f) Forrest Cover Type,$\epsilon = 1.0$

Fig. 6: Comparing $F1$-measure of 3 techniques based on k-skyband tree, kd-tree and quadtree on the two real datasets of NBA player stats and Forest Cover Type when k ranges from 0 to 200 and ϵ ranges from 0.1 to 1.0.

rebounding; while the rest of the players are condensed in the areas along the diagonals from the upper-right corners to the lower-left corners.

5 Related Work

Early works to ensure privacy of released data were based on syntactic approaches such as k-anonymity [16, 31] and ℓ-diversity [24]. However, these approaches only satisfy syntactic privacy notions, and cannot provide formal guarantees of privacy as differential privacy. Differential privacy ensures that no matter what knowledge or power an adversary has, the adversary cannot infer an individual's presence in a dataset from the randomized output.

In this work, our goal is to perform k-skyband queries under differential privacy. Initial efforts on differential privacy [10–12, 14, 21] focused on the theoretical proof of its feasibility on various data analysis tasks, e.g., histogram [3, 19, 36]. More recent work has focused on practical applications of differential privacy for privacy-preserving data publishing, such as data publishing based on private spatial decompositions. Inan et al. [20] proposed a differentially private technique to build data-partitioning index structures in the context of private record matching, which uses an approximate mean as a surrogate for median (on numerical data) to build kd-tree. Recent works [9,37] also proposed several private spatial decompositions, such as quadtree, kd-tree and PrivTree, building the noisy trees with effective budget allocation strategies. These differentially private data publishing techniques are specifically crafted for answering range count queries. However, synthesizing the dataset based on the spatial decompositions and

applying BNL to compute k-skyband results cannot capture the accurate results. Data synthesis on the partitions whose true counts are zero but becomes positive after adding noise would introduce too much unnecessary noise for k-skyband results. Unlike these approaches generating a tree for answering k-skyband queries with different k values, our technique generates a private tree for each k value (i.e., choosing the ne regions based on k). Our technique optimizes the data decomposition for k-skyband queries and suppresses data synthesis on partitions whose noisy counts become positive from zero with post-processing techniques. Evaluation results demonstrate the superiority of our space decomposition optimized based on k over the general space decompositions proposed by the existing works.

Differentially private cluster analysis has also been studied in prior work. Zhang et al. [38] proposed differentially private model fitting based on genetic algorithms and McSherry [27] introduced the PINQ framework, both of which have been applied to achieve differentially privacy for k-means clustering. Nissim et al. [29] proposed the sample-aggregate framework that calibrates the noise magnitude according to the smooth sensitivity of a function. Their framework can be applied to k-means clustering under the assumption that the dataset is well-separated. Chen et al. [8] proposed several techniques that achieve differential privacy for WaveCluster [32,33], which can capture spatial information to detect clusters with complex shapes, e.g. concave shapes. Leveraging private clustering analysis for computing k-skyband results would easily miss some partitions that contain a small number of k-skyband points, since these partitions are too sparse to form clusters.

Another important line of prior work focuses on privacy-preserving database queries over sensitive data distributed among multiple parties. Recently, the advances in the theory of secure multiparty computations [6, 17, 18] proved that comparison, addition, and multiplication (XOR and AND) can be computed securely with reasonable computation cost. Based on these primitive protocols, a line of research has focused on developing efficient secure multi-party communication protocols for various database queries, such as set operations [4,28], top-k queries [34]. These protocols focus on protecting the privacy of the data among multiple parties and only the final query results are released to the public, while our work presents an approach to release the query results without compromising individual privacy of the individuals. What's more, there are no existing secure protocols for k-skyband queries, and building such protocols is not trivial.

6 Conclusion

In this paper we have addressed the problem of k-skyband queries with differential privacy. We propose a general technique BBS-Priv that accepts any space decomposition tree with differential privacy as input and selective performs data synthesis for interested tree nodes to compute private k-skyband results. To improve the query accuracy, we further devise a new space decomposition tree k-skyband tree specifically optimized for k-skyband queries, which partitions the space based on k other than median value or midpoint of each dimension. We further present a suite of techniques to publish a k-skyband tree satisfying differential privacy, and propose a post-processing technique to improve accuracy by suppressing data synthesis on those partitions whose noisy counts

become positive from zero. In the future, we will investigate under differential privacy other categories of multi-criteria decision making queries, such as top-k dominating queries.

References

1. Http://kdd.ics.uci.edu/databases/covertype/covertype.html
2. Nba players statistics, http://www.hoopsstats.com/basketball/fantasy/nba/playerstats
3. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., Talwar, K.: Privacy, accuracy, and consistency too: A holistic solution to contingency table release (2007)
4. Blanton, M., Aguiar, E.: Private and oblivious set and multiset operations. In: ASIACCS (2012)
5. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE (2001)
6. Cachin, C.: Efficient private bidding and auctions with an oblivious third party. In: CCS (1999)
7. Chen, L., Gao, S., Anyanwu, K.: Efficiently evaluating skyline queries on RDF databases. In: ESWC (2011)
8. Chen, L., Yu, T., Chirkova, R.: Wavecluster with differential privacy. In: CIKM (2015)
9. Cormode, G., Procopiuc, C., Srivastava, D., Shen, E., Yu, T.: Differentially private spatial decompositions. In: ICDE (2012)
10. Dwork, C.: Differential privacy: A survey of results. In: TAMC (2008)
11. Dwork, C., Lei, J.: Differential privacy and robust statistics. In: STOC (2009)
12. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: TCC (2006)
13. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. 9 (2014)
14. Feldman, D., Fiat, A., Kaplan, H., Nissim, K.: Private coresets. In: STOC (2009)
15. Feng, X., Gao, Y., Jiang, T., Chen, L., Miao, X., Liu, Q.: Parallel k-skyband computation on multicore architecture. In: APWeb (2013)
16. Ghinita, G., Zhao, K., Papadias, D., Kalnis, P.: A reciprocal framework for spatial k-anonymity. Inf. Syst. 35(3) (2010)
17. Gordon, D.S., Carmit, H., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: STOC (2008)
18. Harnik, D., Naor, M., Reingold, O., Rosen, A.: Completeness in two-party secure computation: A computational view. In: STOC (2004)
19. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially private histograms through consistency. PVLDB 3 (2010)
20. Inan, A., Kantarcioglu, M., Ghinita, G., Bertino, E.: Private record matching using differential privacy. In: EDBT (2010)
21. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? In: FOCS (2008)
22. Kodama, K., Iijima, Y., Guo, X., Ishikawa, Y.: Skyline queries based on user locations and preferences for making location-based recommendations. In: Int'l Workshop on Location Based Social Networks (2009)
23. Levandoski, J.J., Mokbel, M.F., Khalefa, M.E.: Preference query evaluation over expensive attributes. In: CIKM (2010)
24. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data 1(1) (2007)

25. Magnani, M., Assent, I., Mortensen, M.L.: Taking the big picture: representative skylines based on significance and diversity. VLDB J. 23(5) (2014)
26. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: FOCS (2007)
27. McSherry, F.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. Commun. ACM 53(9) (2010)
28. Michael J. Freedman, K.N., Pinkas, B.: Efficient private matching and set intersection. In: EUROCRYPT (2004)
29. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: STOC (2007)
30. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. 30(1) (2005)
31. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In: IEEE Security & Privacy (1998)
32. Sheikholeslami, G., Chatterjee, S., Zhang, A.: Wavecluster: A multi-resolution clustering approach for very large spatial databases. In: VLDB (1998)
33. Sheikholeslami, G., Chatterjee, S., Zhang, A.: Wavecluster: A wavelet-based clustering approach for spatial data in very large databases. The VLDB Journal 8(3-4) (2000)
34. Vaidya, J., Clifton, C.: Privacy-preserving top-k queries. In: ICDE (2005)
35. Valkanas, G., Papadopoulos, A.N., Gunopulos, D.: Skydiver: a framework for skyline diversification. In: EDBT (2013)
36. Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G., Winslett, M.: Differentially private histogram publication. VLDB J. 22(6) (2013)
37. Zhang, J., Xiao, X., Xie, X.: Privtree: A differentially private algorithm for hierarchical decompositions. In: SIGMOD (2016)
38. Zhang, J., Xiao, X., Yang, Y., Zhang, Z., Winslett, M.: Privgene: Differentially private model fitting using genetic algorithms. In: SIGMOD (2013)

A Algorithm of k-skyband tree

Algorithm 2 k-skyband tree

Input: A region r, k in k-skyband queries, the max height of the tree h
Output: A spatial decomposition tree T

1: Q.enqueue(r)
2: **while** Q is not empty **do**
3: n = Q.dequeue()
4: **if** isLeaf(n, h) **then**
5: **continue** // back to Line 2
6: **if** $n.count > k + 1$ **then**
7: N = splitByK($n, k, -1$) // -1 means no noise added
8: **if** $N.ne = n$ **then**
9: N = splitByMidPoint(n)
10: **else**
11: N = splitByMidPoint(n)
12: Q.enqueue($N.ne, N.nw, N.se$)
13: **if** $N.ne.count \leq k$ **then**
14: Q.enqueue($N.sw$)
15: **return** N

Algorithm 2 shows the detailed steps of generating k-skyband tree. Given a region $\langle(x_{min}, x_{max}), (y_{min}, y_{max})\rangle$, k-skyband tree first inserts the input region node r into a queue Q (Line 1), and then removes a node n from Q for splitting (Line 3). If the height of n reaches the maximum height h, n is considered as a leaf node (Line 4) and k-skyband tree continues to process a new node from Q (back to Line 2). If the point count of n is larger than $k + 1$, k-skyband tree uses a function $SplitByK$ (Algorithm 3) to choose a splitting point $s = (s_x, s_y)$ based on k (Line 7), such that the upper-right region $ne = \langle(s_x, s_y), (x_{max}, y_{max})\rangle$ contains more than k data points. When $SplitByK$ cannot find such a ne (i.e., ne is the same as n at Line 8), k-skyband tree uses the midpoint of each dimension as the splitting point (same as quadtree) (Line 9). If the point count of n is smaller than $k + 1$ (not possible to find ne whose point count is larger than $k + 1$), k-skyband tree also uses the midpoint of each dimension as the splitting point (same as quadtree) (Line 11). After splitting, the dominance region of s, $sw = \langle(x_{min}, s_x), (y_{min}, s_y)\rangle$, is considered as a leaf node and no further split is required. The splitting terminates when there are no more nodes in Q to be split (Line 2).

To obtain an upper-right region with more than k points, we propose an efficient algorithm shown in Algorithm 3. Given a region n, the algorithm uses two max heaps (H_x and H_y) to sort the data points within n (Line 2), where H_x (H_y) sorts the points based on their x (y) coordinates. The algorithm accesses a point n_x from H_x and a point n_y from H_y, and uses the x coordinate of n_x and the y coordinate of n_y as a new splitting point $s = \langle s_x, s_y\rangle$(Line 8). Then n_x and n_y are put into a set P (Line 9), which are later checked to see whether they fall into the upper-right region ne split based on s. The reason is that the y (x) coordinate of n_x (n_y) may be smaller than s_y (s_x), and thus n_x (n_y) may not fall into ne. To compute the point count of ne, we only need to check the points in P: each $c \in P$ is checked to see whether it falls into ne (Lines 10-12). If so, then c is moved from P to S, which holds all the points that dominate the current split point (Line 12); otherwise, c remains in P and will be checked again when a new splitting point is formed in the next iteration. Also, due to the way split points are generated, if a point dominates an early split point, it will also dominate later ones. That is why we can safely put that point into S. The algorithm terminates when S contains more than k points (Line 4), i.e., splitting by s guarantees that the upper-right region contains more than k points. The non-private k-skyband tree directly returns the split regions (Lines 16-17) and skip the steps for obtaining the private values for s (Lines 14-15).

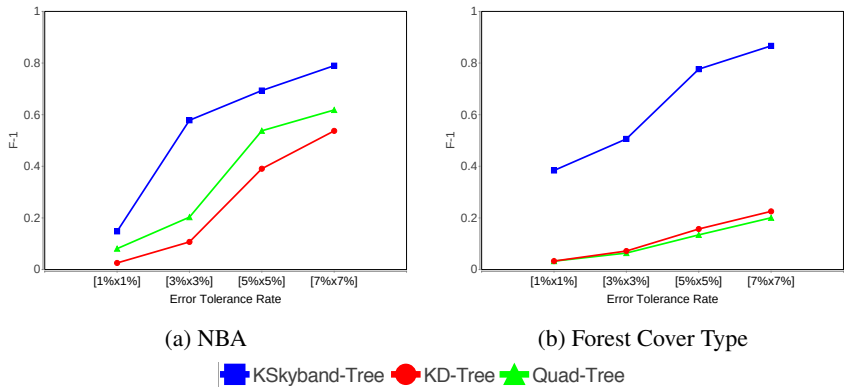
B Results with Different Error Tolerance Rates

A real skyband point is considered to be hit by a private skyline point if the private skyband point is close enough to the real skyband point. Error tolerance rates quantitatively define how close they should be in order to be considered as a hit. Larger error tolerance rates mean more loose constraints on the distance between the private and the real skyband points, and thus make $F1$-measure become better. In other words, when the error tolerance rates become larger, the privacy technique provides better guarantee

Algorithm 3 SplitByK

Input: A region n, k in k-skyband queries, ϵ_s privacy budgets for splitting
Output: A set N that contains four children nodes sw,se,nw,ne

$S = \emptyset$, $P = \emptyset$
H_x.insert($n.data$), H_y.insert($n.data$)
$s_x = n.x_{max}$, $s_y = n.y_{max}$
while $S.size < k + 1$ **do**
  if H_x is empty *or* H_y is empty **then**
    break
  $n_x = H_x$.remove(), $n_y = H_y$.remove()
  $s_x = n_x.x$, $s_y = n_y.y$
  P.add(n_x), P.add(n_y)
  for $c \in P$ **do**
    if $c.x \geq s_x$ **and** $c.y \geq s_y$ **then**
      S.add(c), P.remove(c)
if $\epsilon_s > 0$ **then**
  s_x = EM(s_x,ϵ_s)
  s_y = EM(s_y,ϵ_s)
$N = \{sw, se, nw, ne\}$ = n.split(s_x,s_y)
return N



(a) NBA (b) Forest Cover Type

Fig. 7: $F1$-measure of 3 techniques based on k-skyband tree, kd-tree and quadtree on the real datasets for $k = 40$, $\epsilon = 1.0$, and error tolerance rate ranges from 1% to 7% on each dimension.

in utility. We compute results of $F1$-measure by varying error tolerance rates to observe the impacts of error tolerance rates on the performance.

Figure 7 shows the $F1$-measure results for the two real-world datasets when the error tolerance rate t_x and t_y ranges from 1% to 7%. In each figure, the x-axis shows different values for the error tolerance rates and the y-axis shows the values of $F1$-measure. Due to space limit, we choose to show the results with k set to 40 and ϵ set to 1.0 as the representative results. Clearly, k-skyband tree performs much better than the other two trees in all the error tolerance rates, and its $F1$-measure improves significantly when t_x and t_y reach 7%.