Comparing Synopsis Techniques for Approximate Spatial Data Analysis

A. B. Siddique
Department of Computer Science and Engineering
University of California, Riverside
msidd005@ucr.edu

Ahmed Eldawy
Department of Computer Science and Engineering
University of California, Riverside
eldawy@ucr.edu

Vagelis Hristidis
Department of Computer Science and Engineering
University of California, Riverside
vagelis@cs.ucr.edu

ABSTRACT

The increasing amount of spatial data calls for new scalable query processing techniques. One of the techniques that are getting attention is *data synopsis*, which summarizes the data using samples or histograms and computes an approximate answer based on the synopsis. This general technique is used in selectivity estimation, clustering, partitioning, load balancing, and visualization, among others. This paper experimentally studies four spatial data synopsis techniques for three common data analysis problems, namely, selectivity estimation, k-means clustering, and spatial partitioning. We run an extensive experimental evaluation on both real and synthetic datasets of up to 2.7 billion records to study the trade-offs between the synopsis methods and their applicability in big spatial data analysis. For each of the three problems, we compare with baseline techniques that operate on the whole dataset and evaluate the synopsis generation time, the time for computing an approximate answer on the synopsis, and the accuracy of the result. We present our observations about when each synopsis technique performs best.

PVLDB Reference Format:
A. B. Siddique, Ahmed Eldawy and Vagelis Hristidis. Comparing Synopsis Techniques for Approximate Spatial Data Analysis. *PVLDB*, 12(11): 1583-1596, 2019.
DOI: https://doi.org/10.14778/3342263.3342635

1. INTRODUCTION

The amount of spatial data is exponentially increasing: there are 2.5 exabytes of daily-produced data [37], of which $60-80\%$ is geo-referenced [16, 20]. Space telescopes broadcast [33] 140 GB of data weekly, while $8,000$ tweets are sent every second [34]. A wide range of applications is built on top of these datasets, such as event detection [49], brain simulation [44,51], climate studies [23], trending keywords measurement [35, 43] and so on. To support these applications, selectivity estimation [4, 5, 13, 14, 45, 48], clustering [9], spatial partitioning [21, 22, 40, 54], load balancing [6, 41], query optimization [7], and visualization [47], among others, are common research problems for big spatial data.

In the era of big data, the appetite for algorithms which can provide a fast response to queries on such datasets has grown more than ever. Processing such gigantic datasets takes significant time, even if in-memory big data processing systems, e.g., Apache Spark [59, 60], are used to run approximate algorithms. One research direction to scale algorithms for big datasets is data synopsis-based methods, e.g., using sampling, and run the algorithm on that sample. Many approximation synopsis-based techniques have been proposed, such as Sequential sampling [29], MHist [48], GENHIST [28], Min-Skew [4], STHoles [12], self-tuning histograms (STH) [1], quantiles summaries [25], wavelet transform [45, 53], and so on.

Yet, to the best of our knowledge, *there exists no experimental evaluation which compares different spatial synopsis-based techniques in a common environment*, due to two challenges. First, each data synopsis has a different representation and creation parameters, e.g., sampling ratio or the number of histogram cells, which makes it hard to compare their performance. Second, while existing algorithms can be used as-is with samples, other synopsis-based methods, such as histograms, might require major modifications to the algorithms to work. *The primary contribution of this paper is comparing several important representatives of synopsis-based techniques on three common spatial data analysis problems, in a common environment.* The synopsis techniques that we consider are selected based on their popularity and their coverage of other techniques, as summarized in Table 1, and discussed in Section 2, while the three analysis problems are selected for being fundamental operations that are sometimes used as building blocks to solve more complex problems. Clearly, this study could be expanded in the future, given the huge amount of techniques available in the literature.

Figure 1 overviews the experimental setup of the paper. Specifically, we experimentally compare existing and new synopsis-based methods to solve three fundamental analysis problems: selectivity estimation, k-means clustering, and spatial partitioning on big spatial data. First, we compute four types of data synopses, namely, random and stratified samples, uniform and non-uniform histograms. To make the synopses comparable, we define a parameter B which indicates the memory budget that we can use for the syn-

Proceedings of the VLDB Endowment, Vol. 12, No. 11
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3342263.3342635

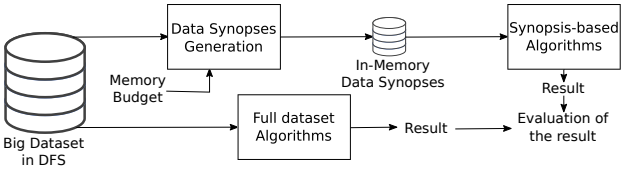


Figure 1: Common environment for data Synopsis-based algorithms evaluation.

opsis. Then, we execute an algorithm on each data synopsis to provide an approximate answer to three popular and diverse problems: selectivity estimation, k-means clustering, and spatial partitioning. To evaluate their accuracy, the synopsis-based algorithms are compared to *full dataset algorithms* that work on the entire dataset.

Previous work has proposed algorithms that operate on sample-based synopses, except for selectivity estimation where histograms have also been used. To achieve a more comprehensive evaluation, a secondary contribution of this paper is that we propose new algorithms that operate on histograms to solve the problems of K-means clustering and spatial partitioning.

We use well-accepted measures to compute the quality of the approximate answers. In specific, for selectivity estimation, we use the relative error as compared to the exact answer, for k-means clustering, we use the sum of squared distances, and for spatial partitioning, we use five well-known quality attributes inspired by the R*-tree index [21]. As shown in Figure 1, these quality measures are used to compare to the algorithms that operate on the full dataset. Based on this comparison, we make observations on when synopsis-based algorithms might be a good choice.

While no synopsis-based algorithm works best for all the given problems, and datasets, we are able to provide *guidelines* for choosing the most appropriate synopsis technique for each of the three problems, given the dataset properties and user preferences. Specifically, we found that for the selectivity estimation, non-uniform histogram-based method proves to be very efficient for selectivity ratios ≥ 0.001 for datasets, when rectangles tend to be small or medium-sized. For k-means clustering, we found that non-uniform histograms do not work well since they distort distance calculations. For spatial partitioning, we found that the underlying partitioning scheme drives the overall quality far more than the memory budget does.

Specifically, our contributions in this work are as follows:

1. We experimentally evaluate two types of samples and two types of histograms for three common spatial problems. We use both real and synthetic datasets of up to 2.7 billion records and 100 GB of data. We vary the memory budget for synopses and study its effect on both the execution time and the quality of the results.

2. As a secondary contribution, to fill the gaps when no algorithms have been proposed for a synopsis-problem combination, we introduce new algorithms to execute K-means clustering, and spatial partitioning using histograms.

3. We provide guidelines on when to use each synopsis technique based on their strengths and weaknesses, given the dataset properties and user preferences.

The remainder of the paper is organized as follows. Related work is discussed in Section 2. Section 3 provides an overview of the experimental study. Sections 4, 5, and 6 cover the details of selectivity estimation, k-means clustering, and spatial partitioning, respectively. Section 7 gives an extensive experimental evaluation and highlights the key findings. Finally, Section 8 concludes the paper.

2. RELATED WORK

Data synopsis is a statistical technique that provides a summarized description, called *synopsis*, of a big dataset, e.g., a sample, a histogram, a wavelet decomposition, or a sketch. In data management, synopses are used to speed up query processing by providing approximate answers based on the synopsis. Almost all the data processing systems including big data frameworks [30, 60] utilize some kind of synopsis for speed and accuracy trade-offs. Yet, to the best of our knowledge, there exists no experimental evaluation which compares different spatial synopsis-base methods in a common environment. We organize the related synopses into three categories, sampling, histogram, and other synopsis methods.

Sampling-based methods: Sampling-based methods are pervasive. BlinkDB [5] utilizes dynamic sampling to provide approximate query answers to standard relational queries. Smart sampling techniques [46, 52] are shown to outperform random sampling. To approximate the result size of a query, a random sampling-based procedure is proposed in [29]. Moreover, sampling is also used to provide big spatial data partitioning techniques in different systems, such as SATO [54], SpatialHadoop [21, 22], ScalaGiST [40], and Simba [57]. A weighted sample based clustering method is presented in [58] where the probability distribution of the dataset is computed to assign a weight to the sample. An adaptive sample for selectivity estimation is presented in [39]. A wide range of spatial sampling techniques, such as design and model-based sampling, are reviewed in [55].

Histogram-based methods: AQWA [6] builds a histogram for query workload in different regions to generate adaptive workload-aware partitioning. An image clustering algorithm [24] utilizes non-uniform histograms of the data distribution to improve image segmentation. Histogram-based methods are shown to be superior for accurate spatial selectivity estimation [2, 36, 48]. The self-tuning histograms (STH) [1] are built using the feedback from the query execution engine. It does not depend on the data size, so the building cost is very small. STHoles [12], also built without looking at the data, proposed histogram cell nesting, and outperformed STH. Min-Skew [4] is an approximation of the dataset, utilizing the traditional multi-dimensional histograms while minimizing the spatial skew within each histogram cell. GENHIST [28] uses overlapping cells based on the local density of the data to achieve a compact approximation of the distribution. [13, 32] propose a prefix sum method for the histogram, which can be used to answer range queries in constant time. Instead of storing the frequency of data points lying in a cell, each cell contains horizontal and vertical prefix sum up to each cell.

Other synopsis methods: [45, 53] presents a multi-resolution wavelet decomposition to get a space-efficient approximation of the data, and fast approximate answers for selectivity estimation for On-Line Analytical Processing

Table 1: Existing research in utilizing synopsis techniques for the three problems.

Type of Synopsis	Selectivity Estimation	K-Means Clustering	Spatial Partitioning
Random/ Stratified Sampling	BlinkDB [5]	Sample-weighted clustering [58]	SATO [54], SpatialHadoop [22] ScalaGiST [40], Simba [57]
Uniform/ Non-uniform Histogram	Prefix-sum [32]	Image segmentation [24]	Trivial to apply Section 6.2
Euler and other Histograms	AQWA [6], STH [1] STHoles [12], GENHIST [28]	Not trivial to apply	Not trivial to apply
Wavelets	OLAP queries [45, 53] Cardinality estimation [3]	Not trivial to apply	Not trivial to apply
Sketches	Range Queries [19]	Not trivial to apply	Not trivial to apply

Table 2: A summary of the problems, synopses, and algorithms considered in this experimental study.

Problems	Full Dataset	Data Synopsis			
		RS	**SS**	**UH**	**NH**
Selectivity Estimation (SE)	SE-F (Section 4.3)	SE-RS, SE-SS [11]		SE-UH, SE-NH [32]	
K-Means Clustering (KC)	KC-F [9]	KC-RS, KC-SS [8]		KC-RS, KC-SS (Section 5.2*)	
Spatial Partitioning (SP)	SP-F [25]	SP-RS, SP-SS [17, 38]		SP-UH, SP-NH (Section 6.2*), SP-UHP, SP-NHP (Section 6.3*)	

Whereas, RS = Random Sample, SS = Stratified Sample, UH = Uniform Histogram, NH = Non-uniform Histogram.

* New algorithms adapted in this paper.

(OLAP) queries. Wavelets, along with equi-width, and equi-height histograms are used for cardinality estimation in [3]. A wide range of query-specific sketching techniques are discussed in [18] for approximate query processing. In this paper, we only consider general synopsis methods that can be applied to all three of our studied problems, and hence we do not consider wavelets or sketches.

Table 1 summarizes the related work by presenting how the existing synopses have been utilized to solve the three problems that this paper studies. *In our experimental study, we pick widely used synopsis techniques that have been used for all three popular research problems studied in this paper (we include uniform/non-uniform histograms too, as it is trivial to support spatial partitioning as shown in Section 6.2).* Euler and other histograms, as well as wavelets, have not been used for spatial partitioning or K-means clustering, and it is not trivial to adapt them for these problems, so we do not consider them. Similarly, sketches are query-specific, and thus it is not trivial to use them to solve the other problems.

We use two sampling-based methods, namely, random sampling and stratified sampling, and two histogram-based methods, namely, uniform histogram, and non-uniform histogram, and provide a comprehensive evaluation to understand the trade-offs in the different synopsis-based techniques for big spatial data. We study their usage in three popular research problems on big spatial datasets, namely, selectivity estimation, k-means clustering, and spatial partitioning. The experiments evaluate the effect of the synopsis techniques on the three problems from three perspectives, the synopsis generation time, the time for computing an approximate answer on the synopsis, and the quality (or accuracy) of the result. Our previous work [13] considered a subset of the synopsis techniques used in this paper and only the selectivity estimation problem on point datasets. A one-page abstract of this study is provided in [50] but it does not provide any details of the techniques or the experimental results.

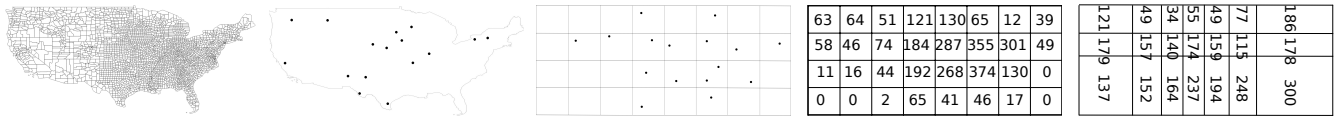
3. OVERVIEW

This section provides an overview of the common environment, which is used to systematically evaluate the performance of the spatial data synopsis for three well-researched problems, i.e., Selectivity estimation, K-Means clustering, and Spatial partitioning. Table 2 summarizes the synopses and problems, which are considered in this experimental evaluation. For a systematic and fair evaluation of the synopsis-based algorithms, we generate a data-synopsis by fixing the allowed memory budget for every synopsis technique. Once, the synopsis is generated, synopsis-based algorithms do not have access to the original big dataset, and are only allowed to use the synopsis to produce the result. The quality of the results of the synopsis-based algorithms is measured using the original big dataset utilizing well-known criteria for each problem, as explained in Section 7. Specifically, we experimentally evaluate four data synopsis techniques and three frequently researched problems on big data to comparatively understand the effect of the synopsis.

The performance of the synopsis-based algorithms is not only compared with one another but also compared against algorithms which utilize the original big dataset instead of a synopsis. The goal is to show any possible performance gain or accuracy loss in synopsis-based algorithms.

3.1 Data Synopsis Calculation

The first step to use synopsis-based processing is to calculate the synopsis which can be shared among different problems and algorithms. Figure 2 shows a big spatial dataset and the four data synopses that correspond to it. The input to this step is a big dataset, a data synopsis method, and a parameter B which represents the desired synopsis size (in bytes), also called the memory budget. The output of this step is a synopsis of the dataset that fits within the size B. The parameter B provides control over the synopsis size irrespective of the input dataset, synopsis technique, and the synopsis-based algorithm. For example, the four data synopses, shown in Figure 2, use a budget of 256 bytes.



63	64	51	121	130	65	12	39
58	46	74	184	287	355	301	49
11	16	44	192	268	374	130	0
0	0	2	65	41	46	17	0

121	49	34	55	49	77	186	
179	157	140	174	159	115	178	
137	152	164	237	194	248	300	

(a) Input dataset **(b)** RS **(c)** SS **(d)** UH **(e)** NH

Figure 2: An input spatial dataset and its four synopses each of size, $B = 256$ bytes.

Fixing the synopsis size enables us to compare the quality and performance of the various synopses irrespective of their creation parameters. It can be set based on the capability of the machine that will run the synopsis-based algorithm, e.g., its available memory size. We have implemented two variants of sampling, i.e., random sampling and stratified sampling, and two variants of the spatial histograms, i.e., uniform and non-uniform histogram.

Random Sample (RS): To get a random sample of size B (Figure 2b), we draw a random sample of an expected fraction $\sigma \in [0, 1]$ in parallel; if the number of records in the input is $|I|$, and p_0 bytes are required to store a sample point, then the sampling fraction is $\sigma = \frac{B/p_0}{|I|}$.

Stratified Sample (SS): To get a stratified sample of size B (Figure 2c), we define the strata as the grid cells of a uniform histogram. That is, if f_i represents the frequency of the cell i, and we want to pick σ fraction of the dataset, then would randomly pick $\sigma \times f_i$ elements from that cell.

Uniform Histogram (UH): The uniform histogram (Figure 2d) is represented by a $2d$ array. For a budget of the size B and a cell entry of size p bytes, the total number of cells is, $d_1 = \lfloor B/p \rfloor$. Assuming square-shaped cells, each cell has an area of $Area(I)/d_1$ and a side length of $c_1 = \sqrt{Area(I)/d_1}$ where $Area(I)$ is the are of the MBR of the input data. This results in a uniform grid size of $L_1 = \lfloor Width(I)/c_1 \rfloor$ columns and $W_1 = \lfloor Height(I)/c_1 \rfloor$ rows. The frequency of each cell is computed based on the centroids of the records in parallel.

Non-uniform Histogram (NH): In non-uniform histogram (Figure 2e), in addition to storing the counter, we also store the width of each column and the height of each row. Assuming that the width and height values take the same space as one entry in the histogram, e.g., eight bytes, then the number of rows and columns for non-uniform histogram is $W_2 = W_1 - 1$ and $L_2 = L_1 - 1$. To compute non-uniform histogram, we first draw a sample of size B and compute MBR of the input dataset. Then, the width/height of each column/row is computed by splitting the space into vertical/horizontal strips where each cell contains roughly the same number of the sample points. Finally, the whole input is scanned in parallel and the frequency of each cell is computed based on the centroid of every geometry; binary search is used to find the column and row for every record.

3.2 Synopsis-based Algorithms

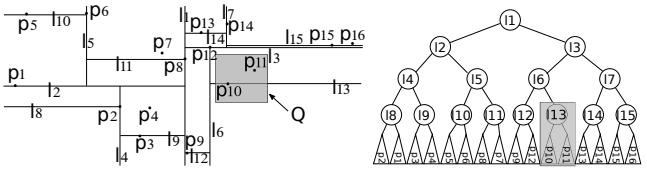
The synopsis-based algorithms operate on the small, fixed-size synopsis, and produce the result by employing the problem-specific logic. For example, this step produces k-cluster centers for k-means clustering, partition boundaries for spatial partitioning, and estimates for the given query ranges based the data-synopsis. Sections 4, 5, and 6 give more details about the algorithms based on the specific synopsis methods. The quality of the results is measured based on the original big dataset. To do this, the result computed by the synopsis-based algorithms is first broadcasted to all the machines in the cluster while the input dataset is scanned in parallel to measure the quality of the results. For example, in K-means clustering, the synopsis-based algorithm produces the k-cluster centers, which are used to measure the quality of the clustering for the whole dataset. The partition boundaries, generated by the synopsis-based algorithms, are used to calculate the quality of the partitions for the whole dataset. Similarly, selectivity estimates for the given query ranges are measured against ground truth for quality evaluation. It is noteworthy that the data synopsis generated for a fixed budget can be reused for all future synopsis-based algorithms.

3.3 Handling Dynamic Data

Dynamic (streaming) spatial data is an important type of data to be handled. Although in this experimental evaluation, all the synopses are computed on the static datasets, we can leverage previous work on incrementally computing synopses on streaming data. For example, sampling over spatial streams proposed in [15, 31] can be used to calculate a random sample of size B. Similarly, histograms can be computed based on the algorithms proposed in [26, 27, 56]. Moreover, combining the UH synopsis calculation with the random sampling, SS synopsis can also be generated. Then, SE-UH and SE-NH can answer the selectivity queries by recomputing the prefix-sum on the updated histogram. SE-RS and SE-SS can work with minor implementation-related change in k-d tree construction, where sampled data is used to divide the space as it comes, instead of dividing the space on the median value. New partitions can be generated or new cluster centers can be found based on the updated synopsis, by applying the respective synopsis-problem combination. We note that adaptive partitioning or adaptive clustering based on the updated synopsis is beyond the scope of this study.

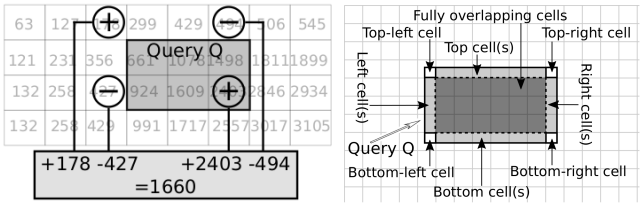
4. SELECTIVITY ESTIMATION (SE)

Selectivity estimation is a well-studied research problem which estimates the number of records in a given query range. This problem has many applications such as approximate query processing and query optimization where the query selectivity helps the query optimizer in deciding which access path to use, i.e., whether or not to use an index. In this study, we provide an estimated number of records in the given query range based on the synopsis. The input to the algorithm is the data synopsis of size B bytes and a rectangular query range Q while the output is a single number that represents the estimated number of records in Q. We use two variants of sampling and the three variants of histogram-based synopsis in this problem.



(a) K-d tree index of sample points **(b)** k-d tree

Figure 3: Sample-based selectivity estimation using K-d tree.



(a) Query answer with Prefix-sum **(b)** Unaligned Query

Figure 4: Selectivity query answer using histogram.

4.1 SE-RS, SE-SS: Estimation on Samples

For sample-based data synopses (either computed through the random sample or stratified sample), we insert the sample points into an in-memory K-d tree [11] to speed up the approximate results for the query range. For n sample points, the K-d tree can be built in $O(n \log n)$ time. We only need to construct the K-d tree index *once per data synopsis* for a fixed memory for all the future selectivity queries. The K-d tree index can efficiently return the number of point in the given query range. To get results for the original dataset, we divide the K-d tree index result by the ratio of the sample. For example, if the sampling ratio in the synopsis step is chosen to be 0.02, then the resulting value from the K-d tree index will be divided by 0.02 to get the approximate results from the original dataset for the given query range. To build the K-d tree, we recursively divide the space over the median value of alternate axes, starting with the x coordinate. For example, Figure 3a shows how to partition the RS synopsis (Figure 2b), where $p_1, p_2, .., p_{16}$ represent points and $l_1, l_2, .., l_{15}$ partitions in Figure 3a. The resulting tree is illustrated in Figure 3b for the same synopsis. Depending on the range query, we need to traverse a small or a big portion of the K-d tree to get the selectivity estimation. For the query Q in Figure 3, we only need to select a small portion of the tree. But, for very big range queries, a big portion of the tree would need to be selected. This means that the size of the query affects the running time as further shown in the experiments in Section 7.

4.2 SE-UH, SE-NH: Estimation on Uniform/ Non-uniform Histograms

For both uniform and non-uniform histograms, the selectivity estimation query locates all the grid cells that overlap the query rectangle Q and sums all their frequencies. The prefix-sum technique [32] is used to compute that sum in a *constant time*. The prefix-sum step runs *only once per histogram-based synopsis* in linear time where it computes the horizontal and then vertical prefix sum over the histogram. As a result, the frequency of each cell will be the total number of points in that cell plus that are to its left or top, which is used to answer all the future selectivity queries. To provide the answer for any given range query, the top-left, and bottom-right corner cells are identified. For the uniform histogram, these cells can be identified in constant time. Whereas, in the case of the non-uniform histogram, these two cells are identified using binary search in the columns and rows. Assuming the *index* for the top-left and bottom-right cells to be i_1, j_1 and i_2, j_2 respectively, the final answer is computed by adding the frequencies of $i_1 - 1$, $j_1 - 1$ and i_2, j_2 and subtracting the i_2, $j_1 - 1$ and $i_1 - 1$, j_2 indices. For example, Figure 4a presents the prefix-sum of the synopsis UH (Figure 2d), a range query (gray), and how to calculate the answer for the range queries in constant time (e.g., $+178 - 427 + 2403 - 494 = 1660$).

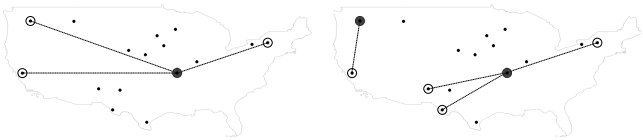
The above technique is accurate only if the query is perfectly aligned with the grid boundaries; otherwise, there might be over- or under-counting depending on how partial cells are handled. To improve the accuracy of the result, the frequency of a partially overlapping cell is scaled down according to the ratio of its area that overlaps the query rectangle Q. For example, if only one-quarter of the cell overlaps Q, its frequency is multiplied by 0.25. This is done under the assumption that the data in each grid cell is uniformly distributed. The selectivity estimation computation can still be done easily in constant time by grouping partially overlapping cells into eight groups depending on their position (left, right, top, bottom, top-left, top-right, bottom-left, bottom-right). For example, Figure 4b presents how the range query can be split into parts. After getting the answer for every portion of the query, the answer is multiplied by the respective fraction it covers the cell of the histogram. Finally, all the values are summed. This calculation is still in constant time irrespective of the size of the range query or whether it is aligned or not with the histogram cell boundaries.

4.3 SE-F: Estimation on Full dataset

The baseline algorithm operates on the original input dataset. It scans the dataset in parallel and filters the records based on the intersection with the given range query. It counts the numbers of filtered records. This algorithm always gives an exact result, but it takes significantly more time than the synopsis-based algorithms, as it has to scan the whole dataset for every query.

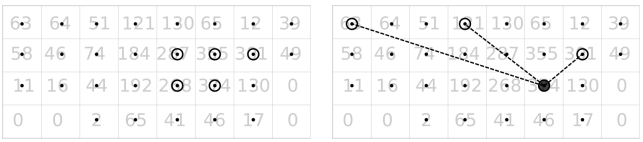
5. K-MEANS CLUSTERING (KC):

K-mean clustering is considered one of the most important unsupervised learning problem, which tries to group objects having some kind of similarity into one cluster. We employ variants of K-means++ [8] to cluster the data into K clusters. The input to the synopsis-based k-means clustering algorithms is a synopsis of size B bytes and the number of clusters K. The synopsis-based algorithms run in an iterative mode until convergence or until the number of iterations reaches a pre-specified upper bound, and return k cluster centers. To compute the quality of the clusters, we use the cluster centers to allocate every record in the whole dataset to the closest center in parallel. In Section 7, we show extensive experiments, which show the superiority of the some of the synopsis-based algorithms, which can outperform scalable K-Means++ [9], as implemented in Apache Spark's MLlib in some situations.



(a) Top candidates for second cluster center (unfilled circles). **(b)** Top candidates for third cluster center (unfilled circles).

Figure 5: Initial cluster centers calculation for the sample.



(a) Top candidates for first cluster center (unfilled circles). **(b)** Top candidates for second cluster center (unfilled circles).

Figure 6: Initial cluster centers calculation for histogram.

5.1 KC-RS, KC-SS: Clustering on Samples

The same synopsis-based algorithm operates on the sample generated through a random sample or stratified sample. The input to this algorithm is the number of clusters K, and a sample of n points (n data points correspond to the memory budget B bytes), such that $n \geq K$. The first initial cluster center is picked uniformly at random, and the remaining $K-1$ initial cluster centers are picked according to the probability distribution $d(x,C)^2$. Where $d(x,C)$ is the Euclidean distance of the record x from the closest cluster center C. Basically, it tries to spread out the initial cluster centers.

For example, Figure 5a shows how the first initial cluster center is picked uniformly random (filled circle in Figure 5a). Squared Euclidean distance of all the remaining points is calculated from it. Thus, the farthest point will have the highest chance to be picked as the second initial cluster center (distance for top three candidates is shown). Since, the algorithm is probabilistic, for a concrete example, we pick one point as the second initial cluster center (filled circle in Figure 5b). Figure 5b shows top candidate points (unfilled circles) with the higher chances to be picked up as the third initial cluster center based on their squared distances from the closest centers. Once, initial cluster centers are selected, then Lloyd algorithm (which recursively assigns points to the closest center, and recalculates the center) is run to obtain final K cluster centers, which are used to cluster the whole dataset into K clusters in parallel. The output of this algorithm is the final locations of the K centers.

5.2 KC-UH, KC-NH: Clustering on Uniform/ Non-uniform Histograms

The input to this variant of K-Means++ is a $2d$ array of size $r \times c$, and the number of clusters K. The basic idea is to represent each cell that has a frequency f, by f points all located at the center of the cell. This is equivalent to creating $r \times c$ points, each located at the center of a cell and is assigned a weight f equal to the frequency of that cell. The first initial cluster center is picked at random, such that the cell with higher weight has a higher chance of being picked as the cluster center. The remaining $K-1$ cluster centers are picked according to the probability distribution $f \times d(x,C)^2$. Where $d(x,C)$ is the Euclidean distance of the record x from the closest cluster center C, and f is the weight for data point x, generated at the cell center. This results in the same behavior as replacing the cell with f points all located at its center. In this way, data points with the higher frequencies, and far away from the existing cluster centers would have a higher chance to be picked as the next cluster center. Figure 6 presents how histogram-based synopsis is used to generate points at the center of the histogram cells with the weight f. Figure 6a shows the top five candidates with higher weights f (unfilled circles) for the first initial cluster center. Again, to be concrete, we pick one of these as the first cluster center (filled circle in Figure 6b), and calculate the squared distances for the rest of points and multiply with the corresponding weight f for the second cluster center. Once the initial K centers are picked, weighted K-Means is used to obtain the final K cluster center. At each iteration, the centers of the K clusters are updated to the weighted centroid of all the cells assigned to this cluster. Almost same algorithm is used for clustering the uniform and non-uniform histogram based synopses. The only difference is that in case of uniform histogram, the centroids of the histograms are calculated in constant time based on their location, and if the frequency is zero for that cell, we simply exclude that cell. Whereas in non-uniform histogram we pick the stored values, and almost no cell would have zero frequency, as the non-uniform histogram strives to distribute equal number of the records in each histogram cell.

To measure the quality of the clusters, we scan the whole dataset in parallel and find the nearest cluster center for each data point, and accumulate the squared error, which is the distance of the record's centroid from the cluster center. The total squared distance is used as a quality measure of the k-means clustering as further detailed in the experiments. In the same scan of the data, records can be assigned to the appropriate cluster.

5.3 KC-F: Clustering on Full dataset

A scalable version of K-means++ [9] is implemented in Apache Spark's MLlib, which utilizes full dataset. It speeds up the initial cluster center selection by parallelizing it. It oversamples by sampling each data point independently and then picking k initial cluster samples from these sampled data points. Finally, it iterates over the entire data multiple times to move the cluster centers according to the original Lloyd's algorithm [42].

6. SPATIAL PARTITIONING (SP)

Spatial partitioning is a widely researched problem for big spatial data which takes a large dataset and partitions it into smaller-sized subsets while *balancing* the sizes of these partitions and maintaining their *spatial locality*. The synopsis-based spatial partitioning algorithms take as input a synopsis of size B, and the desired number of partitions s while the output is a set of at least s MBRs that define the boundaries of each partition. s can also be set based on the size of the input dataset and block size of Distributed File System (DFS). Notice that s is used only as a hint while the partitioning algorithm might generate more partitions. This

(a) Vertical strips **(b)** Horizontal strips

Figure 7: STR Spatial partitioning for the sample.

63	64	51	121	130	65	12	39
58	46	74	184	287	355	301	49
11	16	44	192	268	374	130	0
0	0	2	65	41	46	17	0

(a) Vertical strips

63	64	51	121	130	65	12	39
58	46	74	184	287	355	301	49
11	16	44	192	268	374	130	0
0	0	2	65	41	46	17	0

(b) Horizontal strips

Figure 8: STR Spatial partitioning for the histogram.

flexibility simplifies the implementation and allows for more optimizations as shown below. In order to measure the quality of the partitions, and actually generate the partitions for the whole dataset, it scans the original input dataset in parallel and assigns each record to one of the partitions based on their MBRs.

6.1 SP-RS, SP-SS: Partitioning on Samples

The same synopsis-based algorithm operates on the random and stratified samples. We use STR algorithm [38] and R*-tree [10] as partitioning techniques. The STR algorithm uses a sample of size n points (n data points corresponds to the memory budget B bytes), already computed by the synopsis step, to partition the space into s cells of roughly the same size. First, we compute the degree of the STR tree $g = \lceil \sqrt{s} \rceil$ and run two iterations of the STR algorithm. In the first iteration, the points are sorted by x and split into g vertical strips each containing roughly the same number of points. In the second iteration, each vertical strip is independently sorted by y is split into g partitions of roughly the same size. For example, Figure 7a shows verticals strips to partition the data, and similarly, Figure 7b presents horizontal strips within each vertical strips in such a way that every partition has roughly equal number of records. For R*-tree, we use an open source implementation [17] to build an R*-tree with at least s leaf nodes by setting the maximum leaf node capacity to $M = n/s$.

6.2 SP-UH, SP-NH: Partitioning Uniform/ Non-uniform Histograms

It uses the frequency of the cells to determine the partition boundaries. Each partition contains approximately $|I|/s$ elements, where $|I|$ is the sum of frequencies of all the cells, which is also equal to the total number of records in the input dataset. Similar to the sampling-based algorithm, this algorithm runs in two iterations defining the vertical and horizontal strips, respectively. In the first round, the MBR of the whole input is divided into s vertical strips such that each strip contains roughly $P_v = |I|/s$ elements. This is done by scanning the histogram from left to right while accumulating the total value of each column one-by-one. Once the accumulated value goes beyond P_v, a vertical split line is created at this point and we subtract P_v from the accumulator. This process is repeated until all the columns in the histogram are processed. The second iteration repeats the same process for each vertical strip that was created in the first round. That is, the rows inside each vertical strip are scanned from top to bottom to split them into partitions each containing $P_h = P_v/s$ points. This algorithm works for both uniform and non-uniform histograms. For example, Figure 8 shows how histogram-based synopses (Figure 2d) can be used to generate partition MBRs. Note that the partition boundary has to be aligned with the cell boundaries.

63	64	51	121	130	65	12	39
58	46	74	184	287	355	301	49
11	16	44	192	268	374	130	0
0	0	2	65	41	46	17	0

(a) Vertical strips

63	64	51	121	130	65	12	39
58	46	74	184	287	355	301	49
11	16	44	192	268	374	130	0
0	0	2	65	41	46	17	0

(b) Horizontal strips

Figure 9: Spatial partitioning for the histogram partial cells.

6.3 SP-UHP, SP-NHP: Partitioning on Uniform/ Non-uniform Histograms Partial cells

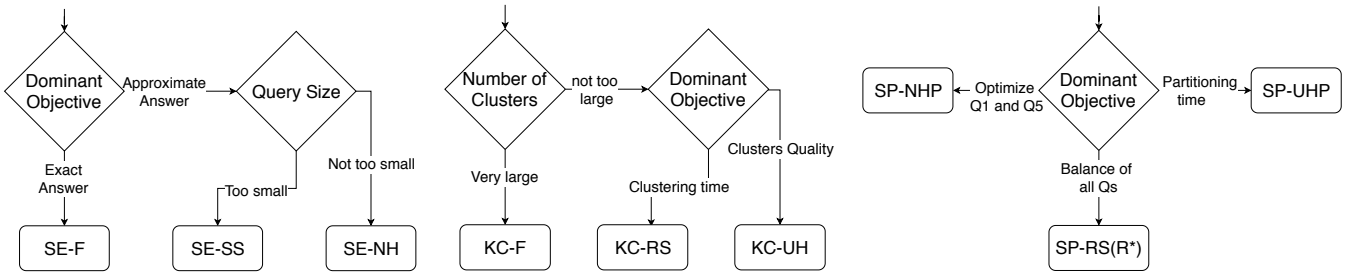
Ideally, each partition should contain approximately $|I|/s$ elements, but in case of highly skewed data, the above algorithm might lead to load imbalance since it has to abide by the column and row boundaries created by the histogram. To overcome this scenario, we can make a slight modification to the algorithm described above. In the first round, instead of placing the vertical split lines at column boundaries, we split the column that causes the overflow (assuming uniform data distribution in that column) such that the created vertical strip contains an expected size of P_v. The same process is done when splitting vertical strips using horizontal lines. Both uniform and non-uniform histograms are split as needed. As shown in Figure 9, we do not need to abide by the histogram cell boundaries, thus more balanced partition MBRs based on the histogram-based synopses (shown in Figure 2d) can be generated.

6.4 SP-F: Partitioning on Full dataset

In Apache Spark, a readily available implementation, called *ApporxQuantile*, is used to compute approximate quantiles. The implementation of the ApproxQuantile is essentially an optimized variation of the Greenwald-Khanna algorithm [25]. If a dataset has N elements, and a quantile at probability q, and up to error rate e is queried, it will return a value r, i.e., $\lfloor ((p-e) \times N) \rfloor \leq rank(r) \leq \lceil ((p+e) \times N) \rceil$. Using this method, we first generate s vertical strips by querying quantiles with respect to x element of the dataset, and then within each vertical strip, we generate the horizontal strips by querying the quantiles with respect to the y elements of the data points within that strip. These horizontal and vertical strips serve as the boundaries of the partitions.

7. EXPERIMENTAL EVALUATION

This section provides an extensive experimental evaluation on the proposed problems using various data synopsis methods.



(a) Selectivity Estimation **(b)** K-Means Clustering **(c)** Spatial Partitioning

Figure 10: Guidelines on when to use which synopsis technique.

Table 3: Dasets

Name	Size	Records	Description
`all-nodes`	96 GB	2.7 billion	Points
`edges`	23 GB	70 million	Polygons
`all-objects`	92 GB	263 million	Mixed
`synthetic`	51 GB	250 million	Rectangles

Table 4: Performance Metrics for each problem

Problem	Quality Measure(s)	Performance Measure(s)
SE	absolute relative accuracy	Query response time
KC	Sum of Squared Error	Clustering time
SP	Q1,Q2,Q3,Q4,Q5	Partitioning time

Table 5: Parameters for Evaluation

Parameter for	Parameters	Range
Synopsis	Memory budget	$10KB$ to $216MB$
SE	Selectivity Ratio	10^{-4} to 10^{-1}
KC	Number of clusters	10 to $1,000$
SP	Number of partitions	algorithm-dependent

7.1 Summary of Results and Guidelines

In our experimental evaluation, no technique performs best for every problem. For that, we provide selection guidelines in Figure 10. The key finding are summarized below.

Selectivity Estimation: SE-F can provide an exact answer, but it also takes significantly more time than any other method to answer the selectivity query. SE-NH can provide better approximate results when the query range is not too small, e.g., $\geq$ 1mile, in slightly more than constant time. Whereas, for very small query ranges, SE-SS can be chosen. It also takes more time to generate SS synopsis, which should be a non-critical issue given that the synopsis only needs to be *generated once* for all the future selectivity queries.

K-Means Clustering: KC-UH can provide better clustering cost than all others including MLlib's scalable K-Means++ implementation. However, if clustering time is the dominant objective, then KC-RS can provide comparable (to KC-F) quality clusters. Finally, if k is very big, e.g., $\geq 10,000$, then KC-F is the most favorable solution.

Spatial Partitioning: SP-NHP can optimize for Q1 and Q5, whereas SP-RS(R*) can provide a good balance for all the quality measures. However, if the dominant objective is partitioning time, then SP-UHP should be considered.

7.2 Experimental Setup

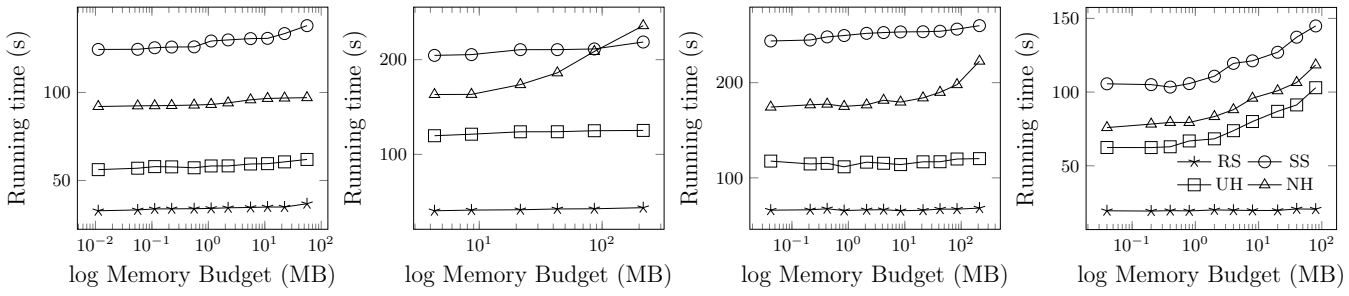
This section describes the setup of our experiments including machine specifications, datasets, parameters, and performance metrics. We use both real [22] and synthetic datasets as listed in Table 3. The MBR of the `synthetic` dataset is $x_1 = -180, y_1 = -90, x_2 = 180, y_2 = 90$. In this MBR, uniformly random points (x and y as double) are generated to be used as the center of the rectangles of width and height of ≈ 2, the rectangles close to the MBR boundaries can have width or height < 2 to keep the centers uniformly distributed and within the MBR.

While any big data system can be used to implement all these algorithms, we are using Apache Spark to exploit its in-memory features, and available implementations, where possible, to make the experiments more *transparent* and *trustworthy*. Experiments on full datasets and synopsis generation are run using Apache Spark 2.1.0 on a 12-node cluster, each with 12 cores, 64 GB RAM, and 10 TB disk storage. They run on CentOS 7.4 and Oracle Java 1.8.0 131. All the synopsis-based algorithms are executed on a machine with 16 cores, 128 GB RAM, and 10 TB HDD.

We generate synopses of the big datasets using a wide range of memory budgets, run selectivity estimation, k-means clustering, and spatial partitioning algorithms for different selectivity ratios, number of clusters, and partitions respectively. To evaluate the effectiveness of each technique, we use quality (listed in Table 4,5) and performance measures, described in the next subsections.

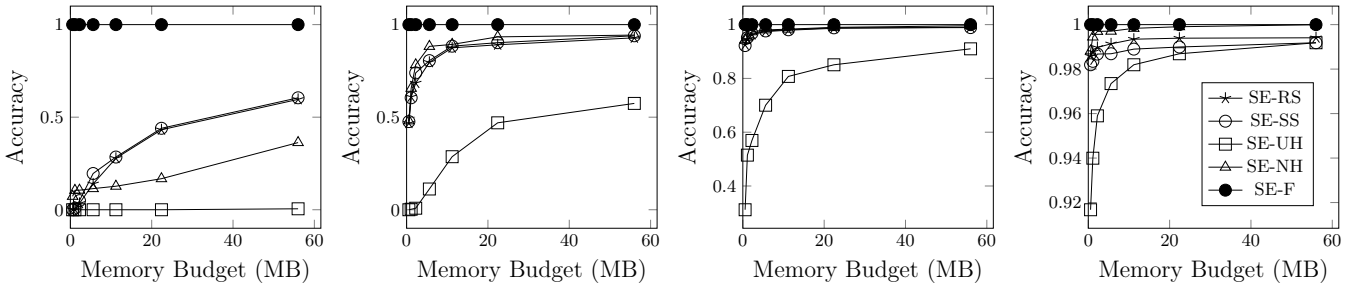
7.3 Synopses Performance

This section provides the performance of the synopsis for the four synopsis methods. Since the synopsis generation step is independent of the synopsis-based algorithms, the experiments in this section are not specific to any problem. In Figure 11, we vary the budget B from 10KB to 216MB, and measure the total running time of the synopsis generation step for each of the four synopsis methods, namely, random sampling(RS), stratified sampling (SS), uniform histogram (UH), and non-uniform histogram (NH). These experiments are executed on the four datasets listed in Table 3. Our experiments show that RS consistently runs faster, whereas SS has the longest running time for `edges`, `all-objects`,



(a) `edges` dataset **(b)** `all-nodes` dataset **(c)** `all-objects` dataset **(d)** `synthetic` dataset

Figure 11: Running time of different synopsis techniques for different datasets.



(a) Selectivity Ratio = 0.0001 **(b)** Selectivity Ratio = 0.001 **(c)** Selectivity Ratio = 0.01 **(d)** Selectivity Ratio = 0.1

Figure 12: `edges`**: Accuracy of all techniques for different selectivity ratios and memory budgets.**

and `synthetic` datasets, as shown in Figures 11a, 11c, and 11d. As SS first computes uniform histograms, then according to the frequency of each cell, the stratified sample is obtained, so it was expected to have more running time. A slightly interesting result is shown in Figure 11b for `all-nodes` dataset, the crossover between SS and NH for $B > 80$MB. The explanation for this crossover is very intuitive; as NH uses binary search to identify the cell for each record, whereas cell identification in UH computation for SS is performed in the constant time. Although `all-objects` and `all-nodes` are roughly similar size datasets, yet the number of records in each vary by a big margin.

The synopsis step is performed only once for a given synopsis method and a memory budget, and all the synopsis-based algorithms can reuse once the synopsis is generated. To conclude this subsection, the running time of all the techniques grow as we increase the budget, B, and the number of the records in the input dataset. Whereas, the running time of the NH is affected the most by the number of records in the input dataset.

7.4 Selectivity Estimation Performance

This section deals with the quality and performance measurement of the selectivity estimation algorithm. To restate, the selectivity estimation queries are answered purely based on the *data synopsis*, and no scan of the big dataset is required. We use `edges` and `synthetic` dataset for evaluation of the different synopsis-based selectivity estimation algorithms. To prepare a query workload, we pick 100 random points from the input dataset and use as query centers. The queries are rectangles with an area of 0.0001, 0.001, 0.01, and 0.1 of the area of the MBR of the input dataset.

7.4.1 SE: Quality Measure

To evaluate the quality of the selectivity estimation algorithms, we use average absolute-relative-accuracy as the quality measure, which is on-the-average how close is the estimate to the ground truth for all the queries. To be able to measure the quality, we use SE-F, which always computes exact answer (we consider it as ground truth) for the given query. For a query q, if the ground truth is t_q and the estimated value is e_q, we compute the accuracy of q as $\max\{0, 1 - |t_q - e_q|/t_q\}$. This gives a range of $[0, 1]$ for the accuracy. The average accuracy of all the 100 queries for each selectivity ratio is used as the accuracy for the corresponding technique.

Figures 12 presents the accuracy of all the selectivity estimation algorithms for `edges` dataset for different selectivity ratios ranging from 0.0001 to 0.1, and memory budgets varying from 1MB to 60MB. In general, the accuracy tends to increase when we increase the budget which is desirable behavior. Particularly, the smaller selectivity ratios, such as 0.001, are more interesting to observe for two reasons, 1) smaller selectivity ratios contain only a small number of records, thus the margin of error is very little, 2) it is closer to most of the real-world scenarios, as generally, users are more interested in smaller region queries, as compared to the large ones (e.g., find restaurants within $5 - 10$ mile as compared to 100 miles.).

For `edges` dataset, SE-NH is more accurate than all other techniques except for very small range queries (e.g., $\leq$ 1mile), where SE-SS and SE-RS have better accuracy. Whereas, SE-UH has the worst accuracy due to sparseness in some areas and denseness at others in the real-world dataset. The main reason for being SE-NH worse than SE-SS and SE-RS for very small selectivity ratios is that since,

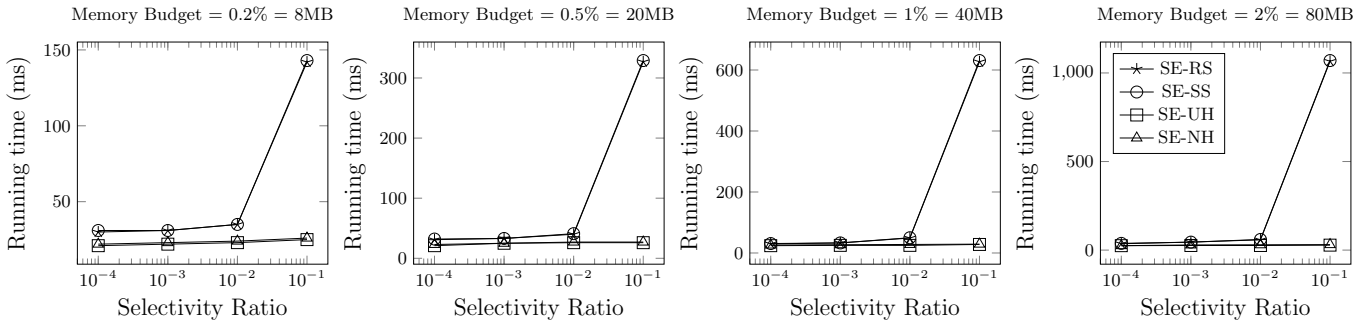
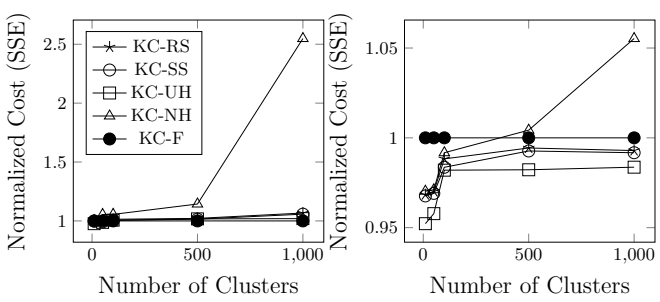
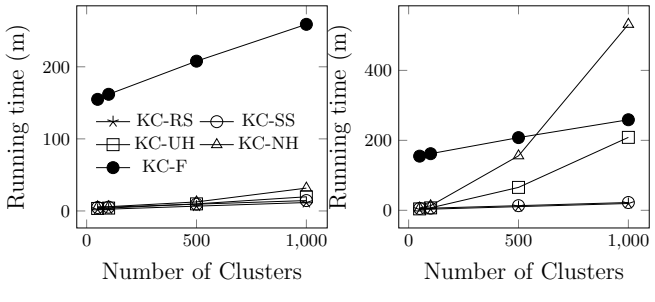


Figure 13: `synthetic`**: Query running time for different selectivity ratios and memory budgets. SE-F** $= 55061ms$



(a) B = 2.16MB **(b)** B = 21.6MB

Figure 14: `all-nodes`**: Cost for clustering techniques.**



(a) B = 2.16MB **(b)** B = 21.6MB

Figure 15: `all-nodes`**: Clustering running time.**

for very small selectivity ratios such as 0.0001, the query size is too small that it mostly covers partial cells of the NH. Moreover, it is also observed that for larger selectivity ratios (e.g., 0.1), all the techniques have accuracy $> 97\%$, because the big query rectangle covers a large number of records, so any technique can easily estimate within 3% error for $B > 10$MB.

7.4.2 SE: Performance Measure

To measure the performance of every technique, we use the average time to answer the estimation query. Figure 13 shows the average time to answer a single selectivity estimation query for the `synthetic` dataset as we vary the budget and the query size. All histogram-based techniques are clear winners as they can answer any query in *constant time*, ascribed to the usage of the prefix-sum technique. Whereas sampling-based selectivity estimation techniques use a k-d tree index to answer the queries. As the sample size expands, the search time will also increase. Similarly, bigger selectivity ratios such as 0.1, influence the running time of the sampling-based method many-fold, as it has to estimate for a very big portion of the tree. Although SE-F always gives the exact answer, yet it takes more than 55 seconds to answer a single selectivity query for `synthetic` dataset, as it has to scan the whole dataset for every query.

To conclude the selectivity estimation results, SE-SS is better for very small selectivity ratios such as, ≤ 0.0001*, and SE-NH outperforms all others for selectivity ratios* ≥ 0.001 *(except SE-F, which is many-fold slower than synopsis-based algorithms) in terms of accuracy for a wide range of experiments, and all histogram-based techniques can answer the selectivity estimation queries in* $O(1)$*, and thus surpass the sampling-based technique. SE-F can only be chosen, when the exact answer is required for very small query ranges, and time to answer is not a consideration.Moreover, precise guidelines on, when to use which technique, are presented in Figure 10a.*

7.5 K-Means Clustering Performance

This section provides the experimental results of both the quality and performance of the k-means clustering problem. For both metrics, we compare to the scalable K-Means++ available in Apache Spark MLlib.

7.5.1 KC: Quality Measure

We use the cost of the clustering as a quality measure which is the Sum of Squared Error (SSE), i.e., the sum of the squared distance of each point to its cluster center. Since the k-means algorithm is randomized, we run each experiment 11 times and report the median of all the costs. In Figure 14, the number of clusters (K) is increased from 10 to $1,000$ for two budget values of 2.16 and 21.6 MB. The cost is normalized by dividing them over the cost of KC-F (MLlib).

For a very small memory budget of 2.16MB (Figure 14a), KC-UH, KC-RS, and KC-SS have comparable quality results to KC-F. For the larger memory budget of 21.6MB (Figure 14b), the KC-UH has the minimum cost, which is even slightly better than KC-F. On the other hand, KC-NH provides a very poor performance for both small and large budgets, especially when $K > 500$. The reason for that poor performance is that it tries to balance the number of records in each cell which produces very tall or wide partitions. Those tall and wide partitions introduce a huge distance error as the points inside each cell might be very far

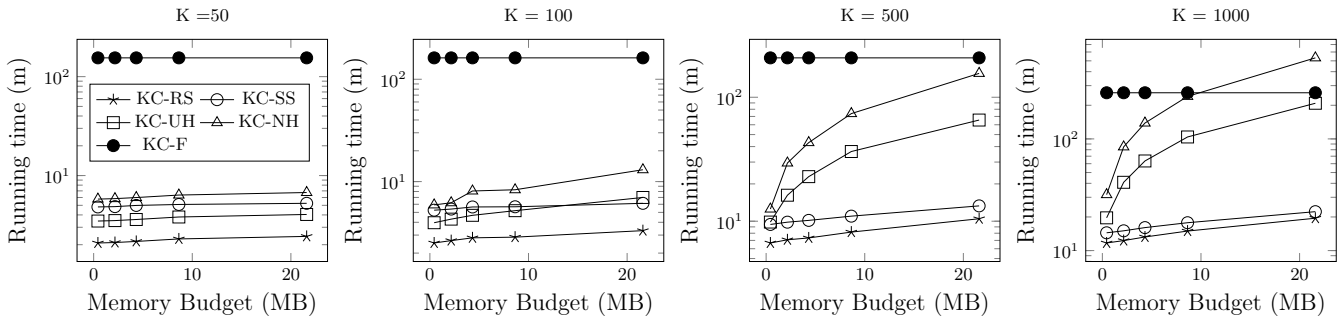


Figure 16: `all-nodes`: Running time of clustering techniques for different budgets and number of clusters.

away from the center. Furthermore, the number of records in the cells are roughly the same in the NH and hence our algorithm cannot utilize these frequencies to influence the cluster center, as it does with UH. Whereas, UH produces square cells of equal size which minimizes the overall distance between the points and their corresponding cell center.

The significance of this result is that for $K \leq 1,000$, synopsis-based algorithms (KC-UH, KC-RS, KC-SS) can achieve comparable or better cluster cost than the state-of-art parallel algorithm for k-means clustering, KC-F, which uses the full dataset. However, when K is very large, then KC-F is an obvious choice. For smaller k, such as $K \leq 1,000$, KC-UH has the least clustering cost, may be opted as the best possible choice. To clarify the computation of the clustering cost, it is computed for the whole dataset, when all the records (in the big dataset) are assigned to the closest clusters centers.

7.5.2 KC: Performance Measure

The running time is used as a performance measure of the clustering problem. Figure 15 shows the running time of the clustering problem for memory budgets 2.16MB and 21.6MB for a wide range of the number of clusters. For the same number of clusters, where KC-UH, KC-RS, and KC-SS had better or comparable clustering cost (as shown in Figure 14) with the KC-F, they are a clear winner in terms of running time, as shown in Figure 15. Figure 16 presents the running time of the k-means clustering for the `all-nodes` dataset. We use four values of K, 50, 100, 500, and 1000. For each one, the budget is changed from 432KB to 21.6MB. The budget does not affect the running time of KC-F which stays constant. For a small value of $K < 100$, synopsis-based techniques outperform KC-F up to two orders of magnitude. However, as K increases, synopsis-based algorithms take more time as they need to run more iterations. Eventually, when both K and B are large, KC-F tends to provide a better running time as it parallelizes all the stages of the algorithm while the synopsis-based methods still have a bottleneck in the k-means clustering step. It is noteworthy that to make a fair comparison with the KC-F, the running time of the synopsis-based algorithms is the sum of the time taken for synopsis generation, the convergence of cluster centers and assignment of all the dataset to the closest cluster centers. Although, synopsis might be reused in a real-world scenario, yet we are considering the case where synopsis is not readily available.

To conclude the results of the k-means clustering, KC-UH can give better cost, whereas the sampling-based techniques can outperform KC-F for running time while providing comparable clustering cost. However, for large K ($\geq 10,000$), KC-F would outperform synopsis-based techniques. Figure 10b presents guidelines for the appropriate technique to use in a given scenario.

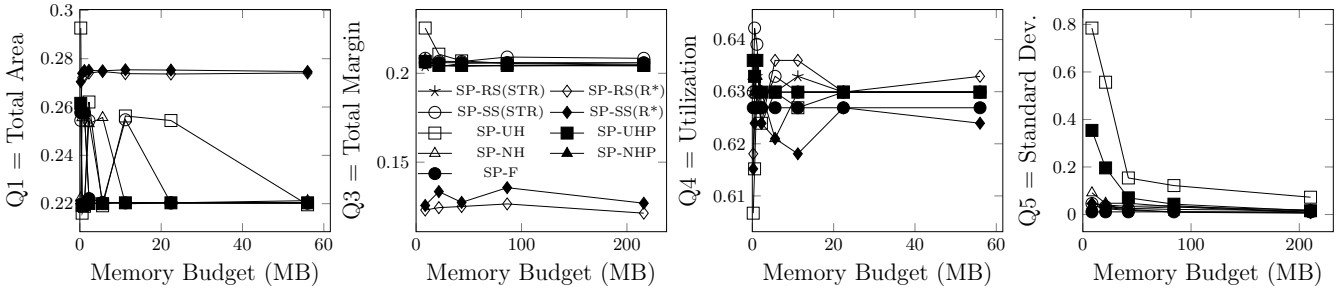
7.6 Performance Measures of Spatial Partitioning

This section provides the experiments on the quality and running time of the partitioning problem. First, we study the quality of the generated partitioning using five standard quality measures [10, 21]. Then, we show the running time of synopsis-based spatial partitioning algorithms.

7.6.1 SP: Quality Measures

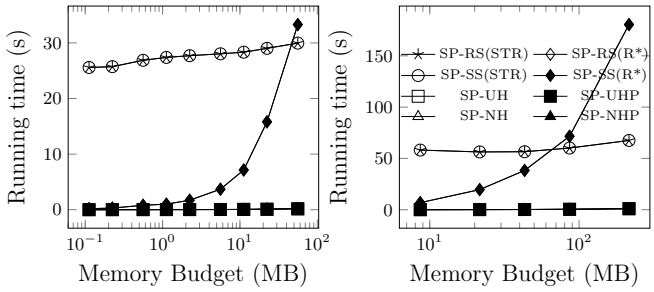
To quantify the quality of the partitions, we calculate five quality metrics of the partitions **Q1** through **Q5** [10, 21]. **Q1** is the sum of the area covered by all the partitions. The lower the value of **Q1** the better the index is, as it indicates more *dead space* is eliminated from the partitions, which did not have any records. **Q2** is the sum of the overlapping area among all pairs of partitions. A smaller value is preferred as it indicates more independence between the partitions. **Q3** is total margin, i.e., the sum of the width and height of all the partitions' MBR. The lower the value the better for the quality as it indicates square-ish partitions. **Q4** measures utilization of the disk space. It is calculated as the ratio between actual data of the partitions to the total capacity of all used blocks of the file system. High Q4 value means higher utilization of the disk space or HDFS blocks. **Q5** is the standard deviation of the sizes of the partition and measures the load balance among all the partitions. Smaller Q5 is the indication of more balanced partitions and thus leading to a good load balance among the machines.

Figure 17 presents various quality measures of the datasets for all the partitioning techniques. As reported in [21], **Q2** is almost zero for all the techniques so we omit the results due to the limited space. Apparently, there is no clear winner that optimizes all **Q**s for all the datasets. For **Q1**, SP-RS(R*), and SP-SS(R*) have the worst quality, whereas, SP-UH and SP-NH have poor quality as compared to the ones SP-UHP, and SP-NHP for smaller memory budgets. However, when the memory budget is 56MB, which is $\approx 2.5\%$ of the input dataset, all partitioning techniques (except R* tree-based) have almost equal performance for **Q1**. On the other hand, R* tree based on samples has almost half



(a) Q1 for `edges` **(b)** Q3 for `all-nodes` **(c)** Q4 for `edges` **(d)** Q5 for `all-objects`

Figure 17: Quality Measures for Spatial Partitioning based on different synopses.



(a) SP-F = $2275s$ for `edges` **(b)** SP-F = $2993s$ for `all-nodes`

Figure 18: Partitioning time for different synopses.

Q3 (margin) value as compared to all other techniques for all the experimented memory budgets because, unlike the STR algorithm, R*-tree has the margin optimization embedded in its algorithm. The utilization (**Q4**) for all the techniques stabilizes at the memory budget of ≈ 21MB and above. R*-tree is able to further improve the utilization when more points are sampled. For **Q5**, SP-UH is performing the worst of all, because of the high skewness in the datasets. We can also notice that SP-UHP improves constantly as the budget increases and at very high budgets it becomes very close to the best quality. Moreover, SP-F is no better than synopsis-based algorithms for all the experiments. In general, as the budget increases, all synopsis-based techniques provide similar quality measures when operating with STR. R*-tree provides a significantly different behavior in which it balances the different quality measures. This suggests that increasing the budget will not improve the spatial partitioning quality as changing the underlying algorithm does. Thus, researchers should invest more in better partitioning algorithm rather than improving the synopsis, at least, for this specific problem.

In summary, we found that splitting the cells for the histogram-based approaches leads to higher quality. Moreover, increasing the budget improves the quality only to some limit but improving the partitioning algorithm has a higher potential in improving the quality. Moreover, SP-F is not performing any better than synopsis-based algorithms. This suggests that future researchers should focus on developing new partitioning techniques.

7.6.2 SP: Performance Measures

The algorithm running time to generate the partition MBRs is used as a performance measure for the partitioning problem. Figure 18 shows the running time of the spatial partitioning problem. Histogram-based partitioning techniques have the least running time, as these only require two passes over the histogram. Whereas, STR based on the samples has to sort the data twice (first by x, then y). R* -tree based on the samples has lesser running time than STR for $B < 50$MB. R* -tree is more affected by the larger memory budgets (> 50MB) because it applies certain heuristics, which takes more time for a bigger sample.

We conclude that there is no clear winner that optimizes all the quality attributes. But, our experiments can provide guidance, when a certain quality attribute is more desirable. SP-F has almost similar results for all the Qs, when compared against synopsis-based methods, whereas it takes significantly more time. Thus, SP-F might not be a good choice for any given scenario. Finally, Figure 10c presents guidelines when to use which spatial partitioning technique.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we present an experimental evaluation of synopsis-based data analysis techniques. For each synopsis, we execute a state-of-the-art algorithm that operates on the synopsis to answer each of the three data analysis problems we consider: selectivity estimation, k-means clustering, and spatial data partitioning. We compare against solving the problems using the full dataset and evaluated the results using well-accepted quality and performance measures. To provide an experimental study, we also filled the gaps by adapting algorithms, where needed. For K-means clustering, we show how K-means++ is applied efficiently on a histogram rather than points.For spatial data partitioning, we show how to extend the existing STR bulk loading algorithm to work with histograms in addition to the sample. While there is no clear winner among the synopsis methods, we provide guidelines to help researchers and practitioners in choosing between those methods. We believe that this paper will open new research directions in extending other problems to work with the various data synopsis.

9. ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation (NSF) under grants IIS-1838222, IIS-1619463, and IIS-1901379.

10. REFERENCES

[1] A. Aboulnaga and S. Chaudhuri. Self-tuning histograms: Building histograms without looking at data. *ACM SIGMOD Record*, 28(2):181–192, 1999.

[2] A. Aboulnaga and J. F. Naughton. Accurate estimation of the cost of spatial selections. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pages 123–134. IEEE, 2000.

[3] I. Absalyamov, M. J. Carey, and V. J. Tsotras. Lightweight cardinality estimation in lsm-based systems. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 841–855, New York, NY, USA, 2018. ACM.

[4] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity estimation in spatial databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, pages 13–24, New York, NY, USA, 1999. ACM.

[5] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42. ACM, 2013.

[6] A. M. Aly, A. R. Mahmood, M. S. Hassan, W. G. Aref, M. Ouzzani, H. Elmeleegy, and T. Qadah. Aqwa: adaptive query workload aware partitioning of big spatial data. *PVLDB*, 8(13):2062–2073, 2015.

[7] W. G. Aref and H. Samet. Optimization for spatial query processing. In *Proceedings of the 17th International Conference on Very Large Data Bases*, pages 81–90. Morgan Kaufmann Publishers Inc., 1991.

[8] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[9] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *PVLDB*, 5(7):622–633, 2012.

[10] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *ACM Sigmod Record*, volume 19, pages 322–331. Acm, 1990.

[11] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[12] N. Bruno, S. Chaudhuri, and L. Gravano. Stholes: a multidimensional workload-aware histogram. In *Acm Sigmod Record*, volume 30, pages 211–222. ACM, 2001.

[13] H. Chasparis and A. Eldawy. Experimental evaluation of selectivity estimation on big spatial data. In *Proceedings of the Fourth International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*, page 8. ACM, 2017.

[14] Y.-J. Choi and C.-W. Chung. Selectivity estimation for spatio-temporal queries to moving objects. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 440–451. ACM, 2002.

[15] E. Cohen, G. Cormode, and N. Duffield. Structure-aware sampling on data streams. In *Proceedings of ACM SIGMETRICS*, pages 197–208. ACM, 2011.

[16] M. . Company. The age of analytics: Competing in a data-driven world — mckinsey & company. https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/the-age-of-analytics-competing-in-a-data-driven-world. (Accessed on 04/21/2018).

[17] S. Contributors. aseldawy/spatialhadoop2 at reindexing. https://github.com/aseldawy/spatialhadoop2/tree/reindexing. (Accessed on 04/25/2018).

[18] G. Cormode. Sketch techniques for approximate query processing. *Foundations and Trends in Databases. NOW publishers*, 2011.

[19] G. Cormode, M. Garofalakis, P. J. Haas, C. Jermaine, et al. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends® in Databases*, 4(1–3):1–294, 2011.

[20] C. Dempsey. Where is the phrase "80% of data is geographic" from? ˜ gis lounge. https://www.gislounge.com/80-percent-data-is-geographic/. (Accessed on 01/10/2018).

[21] A. Eldawy, L. Alarabi, and M. F. Mokbel. Spatial partitioning techniques in spatialhadoop. *PVLDB*, 8(12):1602–1605, 2015.

[22] A. Eldawy and M. F. Mokbel. Spatialhadoop: A mapreduce framework for spatial data. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 1352–1363. IEEE, 2015.

[23] J. H. Faghmous and V. Kumar. Spatio-temporal data mining for climate data: Advances, challenges, and opportunities. In *Data mining and knowledge discovery for big data*, pages 83–116. Springer, 2014.

[24] S. Gao, C. Zhang, and W.-B. Chen. A variable bin width histogram based image clustering algorithm. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, pages 166–171. IEEE, 2010.

[25] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 58–66, New York, NY, USA, 2001. ACM.

[26] S. Guha, N. Koudas, and K. Shim. Data-streams and histograms. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 471–475. ACM, 2001.

[27] S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Transactions on Database Systems (TODS)*, 31(1):396–438, 2006.

[28] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Selectivity estimators for multidimensional range queries over real attributes. *The VLDB Journal*, 14(2):137–154, 2005.

[29] P. J. Haas and A. N. Swami. *Sequential sampling procedures for query size estimation*, volume 21. ACM, 1992.

[30] A. Hadoop. Hadoop, 2009.

[31] J. Hershberger and S. Suri. Adaptive sampling for geometric problems over data streams. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 252–262. ACM, 2004.

[32] C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant. Range queries in olap data cubes. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, SIGMOD '97, pages 73–88, New York, NY, USA, 1997. ACM.

[33] N. HubbleSite. Hubblesite - the telescope - hubble essentials - quick facts. http://hubblesite.org/the_telescope/hubble_essentials/quick_facts.php. (Accessed on 01/10/2018).

[34] Internetlivestats. Twitter usage statistics - internet live stats. http://www.internetlivestats.com/twitter-statistics/. (Accessed on 01/10/2018).

[35] J. Jia, C. Li, X. Zhang, C. Li, M. J. Carey, et al. Towards interactive analytics and visualization on one billion tweets. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 85. ACM, 2016.

[36] J. Jin, N. An, and A. Sivasubramaniam. Analyzing range queries on spatial data. In *icde*, page 525. IEEE, 2000.

[37] M. Khoso. How much data is produced every day? - level blog. http://www.northeastern.edu/levelblog/2016/05/13/how-much-data-produced-every-day/. (Accessed on 01/10/2018).

[38] S. T. Leutenegger, M. A. Lopez, and J. Edgington. Str: A simple and efficient algorithm for r-tree packing. In *Data Engineering, 1997. Proceedings. 13th international conference on*, pages 497–506. IEEE, 1997.

[39] R. J. Lipton, J. F. Naughton, and D. A. Schneider. *Practical selectivity estimation through adaptive sampling*, volume 19. ACM, 1990.

[40] P. Lu, G. Chen, B. C. Ooi, H. T. Vo, and S. Wu. Scalagist: Scalable generalized search trees for mapreduce systems [innovative systems paper]. *PVLDB*, 7(14):1797–1808, 2014.

[41] W. Lu, Y. Shen, S. Chen, and B. C. Ooi. Efficient processing of k nearest neighbor joins using mapreduce. *PVLDB*, 5(10):1016–1027, 2012.

[42] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[43] A. Magdy, L. Alarabi, S. Al-Harthi, M. Musleh, T. M. Ghanem, S. Ghani, and M. F. Mokbel. Taghreed: a system for querying, analyzing, and visualizing geotagged microblogs. In *ACM SIGSPATIAL*, pages 163–172. ACM, 2014.

[44] H. Markram. The blue brain project. *Nature Reviews Neuroscience*, 7(2):153, 2006.

[45] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *ACM SIGMoD Record*, volume 27, pages 448–459. ACM, 1998.

[46] F. Olken and D. Rotem. Sampling from spatial databases. *Statistics and Computing*, 5(1):43–57, 1995.

[47] Y. Park, M. J. Cafarella, and B. Mozafari. Visualization-aware sampling for very large databases. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 755–766, 2016.

[48] V. Poosala and Y. E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *VLDB*, volume 97, pages 486–495, 1997.

[49] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems*, pages 42–51. ACM, 2009.

[50] A. B. Siddique and A. Eldawy. Experimental evaluation of sketching techniques for big spatial data. In *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '18, pages 522–522, New York, NY, USA, 2018. ACM.

[51] F. Tauheed, L. Biveinis, T. Heinis, F. Schürmann, H. Markram, and A. Ailamaki. Accelerating range queries for brain simulations. In *ICDE*, pages 941–952, 2012.

[52] M. Vassilakopoulos and Y. Manolopoulos. On sampling regional data. *Data & knowledge engineering*, 22(3):309–318, 1997.

[53] J. S. Vitter, M. Wang, and B. Iyer. Data cube approximation and histograms via wavelets. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 96–104. ACM, 1998.

[54] H. Vo, A. Aji, and F. Wang. Sato: A spatial data partitioning framework for scalable query processing. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 545–548. ACM, 2014.

[55] J.-F. Wang, A. Stein, B.-B. Gao, and Y. Ge. A review of spatial sampling. *Spatial Statistics*, 2:1–14, 2012.

[56] X. Wang, Y. Zhang, W. Zhang, X. Lin, and W. Wang. Selectivity estimation on streaming spatio-textual data using local correlations. *PVLDB*, 8(2):101–112, 2014.

[57] D. Xie, F. Li, B. Yao, G. Li, L. Zhou, and M. Guo. Simba: Efficient in-memory spatial analytics. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1071–1085. ACM, 2016.

[58] J. Yu, M.-S. Yang, and E. S. Lee. Sample-weighted clustering methods. *Computers & Mathematics with Applications*, 62(5):2200–2208, 2011.

[59] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[60] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.