An Efficient and Effective Algorithm for Density Biased Sampling

Alexandros Nanopoulos
Dept. of Informatics
Aristotle University
Thessaloniki, Greece
alex@delab.csd.auth.gr

Yannis Theodoridis
Dept. of Informatics
University of Piraeus
Piraeus, Greece
ytheod@unipi.gr

Yannis Manolopoulos
Dept. of Informatics
Aristotle University
Thessaloniki, Greece
manolopo@delab.csd.auth.gr

ABSTRACT

In this paper we describe a new density-biased sampling algorithm. It exploits spatial indexes and the local density information they preserve, to provide improved quality of sampling result and fast access to elements of the dataset. It attains improved sampling quality, with respect to factors like skew, noise or dimensionality. Moreover, it has the advantage of efficiently handling dynamic updates, and it requires low execution times. The performance of the proposed method is examined experimentally. The comparative results illustrate its superiority over existing methods.

Keywords

Clustering, Density bias, Sampling, Spatial indexes

1. INTRODUCTION

Sampling is a data reduction scheme that has found broad applications in data mining [10, 8, 21, 24, 19, 11]. For the data mining task of clustering several algorithms capitalize on sampling as means of improving their efficiency [6, 25, 3, 13]. They are based mainly on *Uniform random sampling*, with which every point has the same probability of being included in the sample.

An important example where Uniform sampling is not adequate is the case of datasets with skewed cluster sizes. In this case, small clusters (i.e., with a small number of points) are possible to be missed, since points belonging to them may not be included in the uniform sample. Density Biased Sampling (DBS) has been proposed in [17, 9], with which the probability that a point will be included in the sample depends on the density of its locus. Hence, an adequate number of points from small clusters is included in the sample, due to the increased local density within the clusters.

For instance, let two clusters depicted in Figure 1. The rightmost consists of 50,000 points and the leftmost of 1,000. The density (i.e., the number of points in each cluster with respect to its area) of the small clusters is about 2 times higher than that of the larger cluster. By taking a Uniform sample of 1%, that is 510 points, only 9 points are included from the small cluster. This number is around the expected one, by considering that each point from the small cluster has inclusion probability 1000/51000. When performing DBS (using the SP algorithm that will be described in the following), 87 points from the small cluster are included in the sample. The small cluster, therefore, participates with an adequate number of points in the sample. Thus, while with Uniform sampling the small cluster can be missed (since the very few points can be overlooked as outliers), DBS prevents this problem.

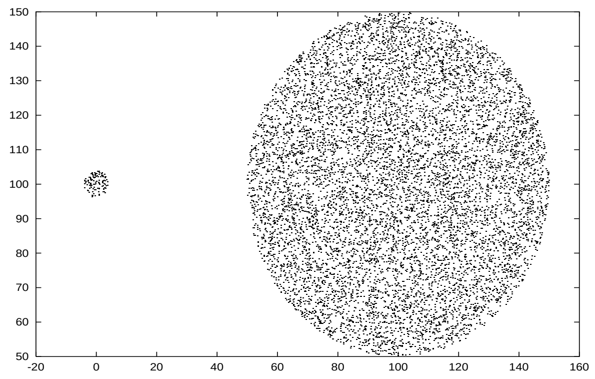


Figure 1: An example of two clusters with different number of points.

Existing DBS algorithms [17, 9] use flat files, by performing one or more database scans and developing estimators for the approximation of local density. Spatial Database Systems exploit access methods for efficient data organization. Sampling from spatial indexes has been introduced by Olken and Rotem [16]. However, they focused on Uniform sampling, which does not consider density-bias. Spatial indexes (like the R-tree [5] and variants) achieve a clustering of objects within their nodes, which preserves the local density information and can provide improved density approximations. Moreover, the index can comprise means for efficiently accessing the examined points.

In this work, we are concerned with answering DBS queries with means of spatial indexes. We describe a new algorithm, which is able to present significant improvements in terms of effectiveness and efficiency. The proposed method is based on the local density information that the nodes of spatial indexes preserve, and on techniques for producing the sample according to the requirements of DBS. With this method:

- Correct clustering results are obtained with smaller sample sizes, which is a significant advantage for reducing the cost of the clustering procedure.
- The sampling quality is preserved with respect to a variety of factors, like skew, noise and dimensionality.
- The exploitation of spatial indexes helps in avoiding the overhead of existing methods in terms of sampling time. Additionally, updates in the dataset are handled locally, thus dynamic data are efficiently addressed.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 contains the outline of the proposed approach. In Section 4, we present the proposed algorithm, whereas Section 5 contains the experimental results both on sampling quality and efficiency. Finally, Section 6 concludes this work.

2. RELATED WORK

Palmer and Faloutsos [17] introduced the problem of non-uniform sampling for clusters with skew sizes, by oversampling smaller clusters and under-sampling larger ones. Since clusters are not known apriori, [17] uses an approximative approach, by dividing the space into equally sized bins. A hash table is allocated, and each table position p corresponds to a group of points g, which contains all points from those bins that are hashed in p. The number, n_g, of the points in each group g is calculated. Each point is included in the sample using the derived groups instead of the unknown clusters. The point-inclusion probability is inversely proportional to n_g (see [17]), thus points from small groups have higher probability of being included. In [17] an efficient algorithm is proposed, which calculates the n_g counters and collects the sample points in a singe database scan. Nevertheless, the method of [17] is significantly affected by the existence of noise, since it tends to over-sample the noisy parts. For this reason, it cannot effectively distinguish between clusters with small sizes and outliers, and it misses clusters. Therefore, it has the drawback of only partially leading to correct clustering results (as indicated by [9] and also verified by results in our work).

Kollios et al. in [9] consider DBS, where a given point is included in the sample according to the local density of the dataset. For the determination of the local density [9] uses kernel density estimation methods. More particularly, for a point x, $f(x)$ denotes the density estimation function. If $f(x) > 1$ ($f(x) < 1$), then the density at x is larger (smaller) than the average density at the entire space. According to this, a point x is included in the sample with probability:

$$P(\text{include point } x) = \frac{b}{k} f^a(x) \tag{1}$$

In Equation 1, b is the expected sample size, $k = \sum_x f^a(x)$ (used for the normalization of probability values so as to be less or equal to one) and a is the tuning parameter.

Experimental results in [9] show that the above method achieves improved sampling quality, compared to [17] and Uniform sampling. Regarding efficiency, the computation of density approximations presents a significant time overhead, since it requires several database scans and high CPU cost. Once the density approximations have been computed, they can be stored and used for DBS, without the need to recompute them. However, in the case of updates in the dataset, the density approximations have to be recomputed for each point. Density approximation in [9] is done with kernel functions, which are applied over a uniform sample of points that is taken from the dataset. Updates may require to take a new uniform sample (since points from the previous sample may have been removed, or new points should be inserted), and compute the new density approximations for each point, paying the aforementioned large cost. Since updates are not handled locally, the efficiency of [9] in handling dynamic data can be impacted.

Olken and Rotem [16] have proposed several algorithms for Uniform sampling from spatial indexes. However, these algorithms do not consider the density-bias. More particularly, the ranked-tree algorithms like the Partial Area R-tree algorithm [16] (including the pseudo-ranked extension of [1]), are based on the iterative location of the k-th entry in the tree, for a random k at each iteration, until the sample is obtained. Evidently, ranked (and pseudo-ranked) algorithms can only be used for Uniform sampling, since they are based on the assumption that each index entry has the same probability of being included in the sample. On the other hand, the paradigm of *acceptance/rejection* (A/R) sampling algorithms [15] can be extended to the case of DBS. For comparison purposes, we have developed extensions of the existing A/R algorithms, which consider density-bias. Their description and performance evaluation is given elsewhere [14], where it is shown that new DBS algorithm, which is described herein, significantly outperforms the extended A/R algorithms in terms of efficiency.

Other related work on Uniform sampling from spatial indexes includes [12, 22]. Finally, [4] proposes the selection of representative points from a dataset and the application of clustering only on them. It describes a focusing technique for the R-tree, which selects the medoid, that is, the most central object of an MBR, from each leaf node. For DBS purposes, from leaves with higher local density, more points should be obtained than from others with lower density. In contrast, [4] selects one point from each leaf. Moreover, small clusters may be confined within a very small number of nodes. By using the above focusing technique, only a small number of points will be selected from small clusters (since one point is selected from each node). This is in contrast to the objective of DBS, which oversamples small clusters based on the increased local density, so as to include an adequate number of points from them, reducing the probability of missing these clusters. Also, the required sample size is a user-defined parameter, and it cannot be restricted by the characteristics of the tree, as in the case of [4] where it is determined by the number of leaf nodes.

3. DBS WITH SPATIAL INDEXES

The update operations in spatial indexes provide a clustering of points within their nodes so as to maximize selectivity during query processing [18, 20]. In the sequel we focus on R-tree [5] and its variants, because they have been broadly used and implemented in commercial and open-source DBMSs (including Oracle, Informix and PostgreSQL). However, an analogous methodology can be followed for other index structures as well. For the R-tree, the clustering of points within nodes is achieved with the split, insertion and deletion operations, which have the objective to preserve proximity between points (i.e., points which are close are likely to be stored in the same leaf) and to minimize dead space. This kind of clustering achieved by the R-tree preserves the local density information within tree nodes.

For instance, Figure 2a depicts two different locations of the space, which correspond to two clusters with different densities. The left location has higher and the right has lower density. In order to maximize proximity and minimize dead space, assuming that no fundamental properties (such as minimum node capacity) are violated, the points from the left location are stored in leaf node A, whereas the points from the right location are stored in leaf node B (in Figure 2a nodes are represented by their Minimum Bounding Rectangles (MBRs)[1]). Consequently, the local density of each point is preserved within nodes A and B, since points are enclosed in the same MBR along with other points from their proximity. In contrast, if the R-tree did not have its clustering properties, the points would have been stored, for instance, in leaf nodes C and D, depicted in Figure 2b. Evidently, in this case no clustering is achieved within nodes and the local density is not preserved.

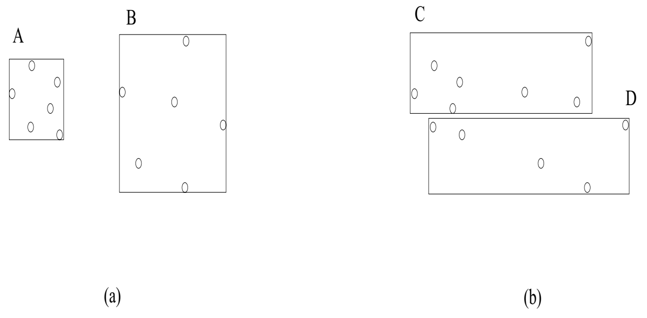


Figure 2: An example of storing two clusters with different densities.

Of course, the R-tree by itself cannot result in perfect clustering, due to its inherent restrictions (the lower/upper limit in the node size and the splitting of a node that overflows into always two nodes). Nevertheless, for the purpose of DBS we are interested in the approximation of local density. The approximation of local density is in accordance with existing DBS methods [17, 9]. Moreover, as indicated by experimental results in Section 5, the proposed approximation is effective and less sensitive to factors like noise and dimensionality, compared to existing approaches.

Let L denote the set of leaf nodes of the R-tree. Since it is desired that points belonging to the same leaf node to have identical probability of being included in the sample [17], we assign the same local density to all points in a leaf. Let f_j denote the fanout of a leaf node j.[2] Consequently, an approximation of a point's local density can be determined by the number f_i of points in leaf i, divided by the volume of i's MBR (for the 2-d space, this corresponds to the area of the MBR). This density value is the same for all points of i and it is denoted as D_i. Hence:

$$D_i = \frac{f_i}{\text{Volume}(\text{MBR}_i)}, \quad i \in L \tag{2}$$

For each leaf i, its density D_i is normalized by dividing it with $D_{\max}$, where $D_{\max} = \max_i\{D_i | i \in L\}$ (in order to normalize the corresponding point-inclusion probability, see Equation 3). Let d_i denote $D_i/D_{\max}$.[3]

Table 1 contains the definition of symbols used henceforth.

Symbol	Definition
d_i	normalized density of leaf i
f_i	fanout of node i
N	number of all data points
L	set of all leaf nodes
a	tuning parameter for DBS
d_{avg}	average value of d_i
s	sample size

Table 1: Symbol table.

4. THE SELECTIVE PASS ALGORITHM

In this section, we present a new method for DBS from an R-tree, called *Selective Pass* (SP). Let N be the total number of points in the tree. SP considers directly the partitioning of the N points into the set L of leaf nodes. SP maintains for each leaf i, its address, the d_i and f_i values. Let this information about leaf nodes be denoted as I_L. SP performs an examination of the members of I_L. For each entry of I_L, which corresponds to a leaf $i \in L$, SP decides whether to include a point from i into the sample according to d_i. If selection is decided, then leaf i has to be fetched from secondary storage and a point is selected at random from i. Otherwise, the reading of i is avoided. To avoid the rereading of leaf nodes, SP assigns to each leaf a number t of trials (t is the same for each leaf). SP is depicted in Figure 3.

The entries of I_L are read from disk during the execution of DBS, where each time only the current page of I_L entries have to remain in main memory (for reasons of fair comparison with other DBS algorithms). The space overhead of I_L is due to $3 \cdot |L|$ numbers that have to be stored, thus it is equal to $O(|L|/B)$ (B is the page size). This single scan over the entries of I_L presents a very small I/O overhead, which does not impact the performance of SP, considering the gainings from avoiding the reading of several leaf nodes (as it is shown in Section 5). It can be easily shown that for, e.g., 5 dimensional data, I_L occupies only 15% of the space required by the upper levels of the index (i.e., excluding the leaves).

[1]In general, a cluster of points may occupy more than one MBRs.

[2]For simplicity, we assume that the internal and leaf nodes have the same maximum fanout.

[3]$D_{\max}$ is maintained by keeping track of the maximum encountered density up to any time point. This maximum value is used even when the corresponding leaf has been deleted, because still all remaining leaves have density smaller than this maximum encountered value.

```
Inputs: The tuning parameter $a$, number of trials $t$, $I_L$
Output: The density-biased sample

SP1.  Start from the first entry of $I_L$, pointing at leaf $i$.
SP2.  Generate a random number $s_i \sim B(t, \frac{f_i}{N} \cdot d_i^a)$.
SP3.  If $s_i > 0$, then fetch $i$.
      Select at random $s_i$ points from $i$.
SP4.  If all $I_L$'s elements are examined, then terminate.
      Else, get the next entry of $I_L$ and execute SP2.
```

Figure 3: The Selective Pass DBS algorithm.

Evidently, for static data the entries of I_L can be obtained during the index creation. For the case of dynamic data, they can be easily accommodated in main memory during updates, considering the large memory sizes today. After an update in a leaf i (insertion/deletion of points), the calculation of the new values for f_i and d_i is performed locally, without affecting the corresponding values for the remaining leaves. In ordinary R-tree implementations, f_i is maintained in i. The calculation of d_i requires f_i and MBR_i, which is also available during the update operation at i. Therefore, SP presents the advantage of handling dynamic updates locally (i.e., without the need to change density estimation in the other leaves). This is in contrast to the algorithm of [9], in which updates in the dataset require the change of density estimation at all points of the dataset. The density approximation of [9] is based on a uniform sample of β points, that may have to be modified after the insertion/deletion of points (due to the deletion of points from the uniform sample or the insertion of new ones that have to be taken into account while forming the kernel functions). Thus, the cost of modifying density estimation for the entire dataset is paid not only for the initial creation of the density estimation function, but also during updates in the dataset.

LEMMA 1. *For each trial of SP, the inclusion probability for a point x, stored in leaf i, is*

$$P(\textit{include point } x | x \textit{ in leaf } i) = N^{-1} \cdot d_i^a \tag{3}$$

Proof. At step SP2, the success probability is $f_i/N \cdot d_i^a$. In case of success, a point x is selected, at step SP3, with probability $1/f_i$. Hence, $P(\text{include point } x | x \text{ in leaf } i) = f_i/N \cdot d_i^a \cdot 1/f_i = N^{-1} \cdot d_i^a$. □

The above lemma shows that, as required by DBS, the point-inclusion probability for each point is biased by the density of the group (i.e., leaf) that it belongs. Another requirement is that the sample size is in accordance with a user-defined parameter.

PROPOSITION 1. *Let $t = M \cdot d_{avg}^{-a}$ be the number of trials assigned to each leaf node (d_{avg} is the average of d_i values among all $i \in L$). Then, with the assumption that each node is of average density, the expected sample size is equal to M.*

Proof. From each leaf j, s_j points are included in the sample, where $s_j \sim B(t, f_j/N \cdot d_j^a)$. The expected number of included points from j is $E(s_j) = t \cdot f_j/N \cdot d_j^a$. If s is the sample size, then $s = \sum_{j \in L} s_j$. Hence, $E(s) = E\big(\sum_{j \in L} s_j\big) = \sum_{j \in L} E(s_j)$. By letting $E(s) = M$ and substituting the value of $E(s_j)$, we get $M = \sum_{j \in L}(t \cdot f_j/N \cdot d_j^a)$. By assuming each node to have the average density, we get $M = t \cdot d_{avg}^a \sum_{j \in L}(f_j/N) = t \cdot d_{avg}^a$. Solving for t, the required equality follows. □

5. PERFORMANCE EVALUATION

To evaluate the performance of the proposed method, we conducted a series of experiments. For comparison, we examine the algorithms of [9] and [17], which are referred as Biased Sampling (BS) and Grid Biased Sampling (GBS) respectively (according to the notation in [9]). We also examine Uniform sampling (US), following the algorithm of [23].

We consider both effectiveness and efficiency. Effectiveness is examined by measuring the quality of sampling with respect to clustering result, along the lines of [17, 9]. We examine the impact of skew in cluster sizes, noise and dimensionality on sampling quality. Efficiency is examined by measuring the execution time with respect to sample size and scalability to the database size.

5.1 Experimental Configuration

We implemented SP in C, the code for BS was provided by the authors of [9] and the code for GBS is available[4]. All experiments were conducted on a Pentium III server at 800 MHz.

The quality of a sample is measured with respect to the number of correctly found clusters. For reasons of fair comparison with prior work, clustering is performed with the CURE algorithm [6]. In order to better control the data characteristics, we used synthetic data. We generated synthetic d-dimensional datasets having k clusters and N points. For each cluster, its center point and radius is specified, and the points belonging to it are generated by following independent normal distributions. We add a specified percentage of noise points (with respect to the number of points belonging to clusters), that follow uniform distribution all over the space of the dataset. The number, NC, of correctly found clusters is calculated by comparing the distances of the centers derived from the clustering algorithm with the actual ones and using a threshold for determining correctness (see [17] for more details).

We tune the a parameter of BS according to the indications in [9]. For datasets containing noise, a was set to 1; in contrast, when no noise exists, a can be set to -0.5 so as to identify very small clusters. For SP, we adopt the same setting for the a parameter. For GBS we used the IRBS variant, according to the results of [17]. The default number of kernels for BS was set to 1000 [9]. The default available memory size for the hash table of GBS [17] and for the buffer space used by SP was set to the 30% of the dataset size. We used the R*-tree [2] variant and the default page size was set to 4 K.

5.2 Results on sampling quality

Clusters with skew sizes

We measured NC (the number of correctly found clusters) for the case of datasets which contain clusters with skewed

[4]at www.cs.cmu.edu/People/crpalmer/dbs-current.tar.gz

sizes. We used 3-dimensional synthetic datasets, which contained 9 clusters (we also examined cases with larger number of clusters, which produced qualitatively similar results that are omitted for brevity). One cluster contained 50,000 points and the remaining ones had 500 points (100 times less). Additionally, 10% noise was added. Also, the number of kernels for BS was set to 3,000 (since the default value of 1,000 produced worst results). The results with respect to the sample size (given as percentage of dataset size) are depicted in Table 2.

Alg.	Sample size					
	0.5%	1%	1.5%	2%	3%	4%
SP	9	9	9	9	9	9
BS	3	6	8	9	9	9
GBS	2	2	3	6	7	7
US	3	4	5	7	8	9

Table 2: *NC* w.r.t. sample size for the case of clusters with various sizes.

BS managed to find the correct clusters earlier than US and it found more correct clusters in the cases where both did not correctly found all clusters. US found the correct clusters only for sample size 4% and larger. GBS did not produce correct results, because it is impacted by the skew in cluster sizes and the presence of noise in the dataset. Thus, it could not effectively distinguish between noise and cluster points. SP clearly outperforms all other algorithms, even at lower sample sizes. Hence, the proposed approach can guarantee correct clustering results with much smaller sample sizes. This is very important, since smaller sample sizes can achieve a significant reduction in clustering execution times.

Noise percentage

To examine the impact of noise, we used analogous datasets to the ones of the previous experiment. We set the sample size to 2% of the dataset size and we varied the amount of added noise. The results are depicted in Table 3, with respect to the amount of noise (given as a percentage).

Alg.	Noise Perc.						
	0%	10%	20%	30%	40%	60%	80%
SP	9	9	9	9	9	9	8
BS	9	9	9	9	9	8	7
GBS	9	8	8	7	5	3	3
US	9	9	7	5	4	4	3

Table 3: *NC* w.r.t. noise percentage.

As expected, the effect of noise is significantly noticeable in the cases of US and GBS. They achieve correct results only when no noise exists (for GBS), or for very small percentage (10% for US). SP clearly presents the best performance. It correctly finds all clusters in all cases with noise less than 80%, and only at 80% they miss one cluster. BS, for very large noise, is affected at a less high value. It starts missing one cluster at 60%, and then, at 80%, two clusters. BS uses approximation with a kernel function, which is based on a uniform sample of β points [9]. As the noise percentage increases, the larger becomes the probability that noisy points to be included in the uniform sample used by the kernel function (a case analogous to US). Noisy points in the uniform sample are included at the expense of not including points from clusters, thus preventing the weight increase around the locus of the latter points. For SP, the approximation is not based on a uniform sample. Noisy points affect only the approximation at the leaf at which they are included (i.e., local effect). This effect can be further reduced by considering heuristics that try to isolate noisy points within leaf nodes. For instance, the density of a leaf node can be determined by forming a core of the most central points (closest to the center of node). We address the examination of such heuristics as further work.

Dimensionality

Next, we considered the impact of dimensionality[5]. Noise was set to 25% and the sample size was 2%. The results are depicted in Table 4.

Clearly, GBS and US do not produce correct clustering results. GBS is mainly affected by the existence of noise. Although one can expect Uniform sampling to be effective for high dimensional data that follow uniform distribution (based on examples from statistic literature), in the examined case it is impacted by the skew in the cluster sizes and the existence of noise.

Alg.	Dimensionality				
	5	8	10	12	15
SP	9	9	9	9	8
BS	9	9	8	8	8
GBS	7	6	5	5	5
US	7	6	5	4	4

Table 4: *NC* w.r.t. dimensionality.

In contrast, SP finds the correct results in all cases with dimensionality less than 15 and only in the latter case it starts missing a cluster. We note here that high dimensionality results into a skew in the densities of the R-tree leaf nodes, where very few nodes have very large density. For this reason we use for normalization the 95 percent density value instead of the maximum value, to address the above situation. BS also finds the correct clusters for lower dimensions, but it is impacted at a less high dimensionality (at 10). Evidently, due to the high dimensionality, the skew in cluster sizes and the noise, BS will require larger sample sizes in order to obtain the correct clustering in these cases; thus burdening the cost of the clustering procedure. Therefore, the proposed approach presents the advantage of being less affected by high dimensionality and producing correct clustering results with smaller sample sizes.

5.3 Results on efficiency

This section reports the experimental results on efficiency. The previous results on quality indicate that US and GBS produce samples that only partially (in limited cases) lead to correct clustering results. For this reason, following the approach of [9], we do not report results on their execution times.

The proposed approach mainly focuses on the exploitation of indexed datasets. This is analogous to the approach

[5]It has to be noticed that, by using CURE, we focus on full-dimensional clustering. For the particularities of very high dimensional spaces, specialized algorithms have been proposed [7].

of [15] for Uniform sampling from datasets for which a B$^+$-tree index is already build[6]. Based on the above, for reasons of fair comparison, we consider an optimization for BS by pre-computing the density estimations and storing them so as to avoid their recalculation during the application of DBS. Evidently, the latter approach is restricted to static data (i.e., it cannot be used when dynamic updates occur), and also it presents a significant space overhead ($O(N/B)$) in order to store the density estimations.

Figure 4a illustrates the results with respect to the dataset size. As shown, BS is clearly outperformed by SP in all cases. BS has to perform one scan over the entire dataset, so as to examine all points in order to decide their inclusion in the sample. In addition one scan over the pre-stored estimations (one for each point in the dataset) is required to compute parameter k (which cannot be pre-computed because it depends on a that is a user-defined parameter – see Equation 1). In contrast, SP performs a selective pass, which avoids the reading of leaves that will not contribute to the sample. Moreover, it has to be noticed that density information for SP is maintained in I_L only for leaves instead of each single point (i.e., $|L| << N$); thus it requires much less retrieval cost.

Finally, we tested the impact of dimensionality on SP and BS. Figure 4b depicts the execution times with respect to the number of dimensions (the sample size was 2%). As previously, SP clearly presents the best performance in all cases.

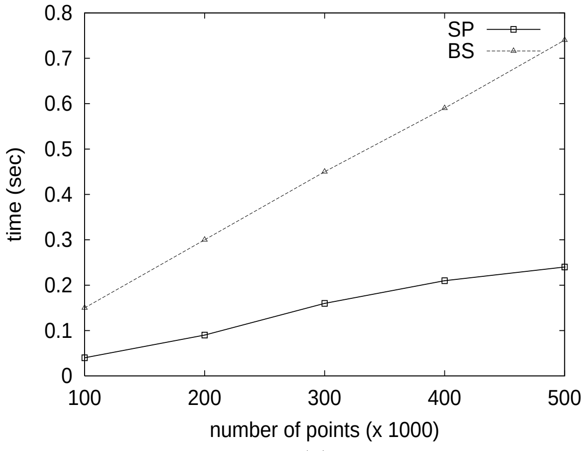
6. CONCLUSIONS

We considered the problem of Density Biased Sampling (DBS) from spatial indexes. Instead of performing DBS over flat files, we exploit the clustering properties of spatial indexes to provide density biased samples of improved quality and with low execution time for the sampling procedure.

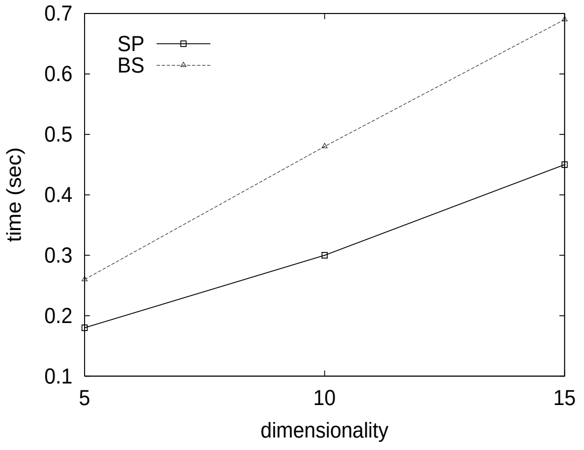
We described SP, a novel DBS algorithm. Its main characteristic is the avoidance of reading nodes that will not contribute to the final sample. Compared to BS, a prior DBS algorithm, SP presents the advantage that updates in the dataset are handled locally, without requiring the rescanning of the entire dataset and recomputing all density approximations. We showed that SP produces samples according to the requirements of DBS.

By using synthetic and real data, the quality of sampling result produced by SP was compared against that of existing algorithms (BS, GBS and US). Our experiments considered a variety of factors, i.e., datasets containing clusters with various sizes, noise and dimensionality. The results show that SP provides correct results, in contrast to US and GBS, which produce correct results only in limited cases. Also, we verified the conclusions of [9], stating that BS attains significantly better samples than US and GBS in terms of quality. However, our results indicate that SP clearly compares favorably to BS. SP is not based on a uniform sample like BS, which bases its approximation on a kernel function that is used with the points in this uniform sample. In SP, the contents of the index nodes can produce improved density approximation based on the local information within nodes.

[6]Otherwise, single-pass algorithms [23] may be the preferred option for Uniform sampling, since the building of the B$^+$-tree will require at least one database pass in addition to the time for the sampling itself.



(a)



(b)

Figure 4: Comparison of SP and BS.

Also, we examined experimentally the sampling time. We considered data for which an index is build. Thus, for fair comparison, we developed an optimized form of BS, which uses pre-computed density approximations. The experimental results show that SP clearly performs better than BS, since BS has to examine each point in the dataset, in addition to the examination of the entire collection of pre-computed density approximations. In contrast, the selective pass of SP avoids reading those that will not contribute to the sample.

Finally, we argue that the proposed approach is easily extendible to the task of outlier detection using samples [9], since in this case sampling with probability *inversely* proportional to density is required. We address this issue as future work.

7. ACKNOWLEDGEMENTS

We would like to thank Prof. Kollios for providing source code for BS.

8. REFERENCES

[1] G. Antoshenkov: "Random Sampling from Pseudo-Ranked B$^+$ Trees". *Proc. Int. Conf. on Very Large Databases (VLDB'92)*, pp.375-382, 1992.

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger: "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles". *Proc. ACM Int. Conf. on Management of Data (SIGMOD'90)*, pp. 322-331, 1990.

[3] M. Breunig, H.-P. Kriegel, P. Kroger, J. Sander: "Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering". *Proc. ACM Int. Conf. on Management of Data (SIGMOD'01)*, pp.79-90, 2001.

[4] M. Ester, H.-P. Kriegel, X. Xu: "Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification". *Proc. Int. Symp. on Large Spatial Databases (SSD'95)*, pp. 67-82, 1995.

[5] A. Guttman: "R-trees: a Dynamic Index Structure for Spatial Searching". *Proc. ACM Int. Conf. on Management of Data (SIGMOD'84)*, pp.47-57, 1984.

[6] S. Guha, R. Rastogi, K. Shim: "CURE: an Efficient Clustering Algorithm for Large Databases". *Proc. ACM Int. Conf. on Management of Data (SIGMOD'98)*, pp.73-84, 1998.

[7] A. Hinneburg, D. Keim: "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering". *Proc. Int. Conf. on Very Large Databases (VLDB'99)*, pp.506-517, 1999.

[8] G. John, P. Langley: "Static Versus Dynamic Sampling for Data Mining". *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)*, pp.367-370, 1996.

[9] G. Kollios, D. Gunopoulos, N. Koudas, S. Berchtold: "Efficient Biased Sampling for Approximate Clustering and Outlier Detection in Large Datasets". *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, to appear, 2002.

[10] J. Kivinen, H. Mannila: "The Power of Sampling in Knowledge Discovery". *Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS'94)*, pp.77-85, 1994.

[11] S. Lee, D. Cheung, B. Kao: "Is Sampling Useful in Data Mining? A Case in the Maintenance of Discovered Association Rules". *Data Mining and Knowledge Discovery*, Vol. 2, No. 3, 233-262, 1998.

[12] C. Lang, A. Singh: "Modeling High-Dimensional Index Structures using Sampling". *Proc. ACM Int. Conf. on Management of Data (SIGMOD'01)*, pp. 389-400, 2001.

[13] A. Nanopoulos, Y. Theodoridis, Y. Manolopoulos: "C^{2}P: Clustering based on Closest Pairs". *Proc. Int. Conference on Very Large Databases (VLDB'01)*, pp.331-340, 2001.

[14] A. Nanopoulos, Y. Theodoridis, Y. Manolopoulos: "Density Biased Sampling with Spatial Indexes". Tech. Report, available at www-de.csd.auth.gr/dbs.pdf, 2002.

[15] F. Olken, D. Rotem: "Random Sampling from B$^+$-Trees". *Proc. Int. Conference on Very Large Databases (VLDB'89)*, pp.269-277, 1989.

[16] F. Olken, D. Rotem: "Sampling from Spatial Databases". *Proc. IEEE Int. Conf. on Data Engineering (ICDE'93)*, pp.199-208, 1993.

[17] C. Palmer, C. Faloutsos: "Density Biased Sampling: an Improved Method for Data Mining and Clustering". *Proc. ACM Int. Conf. on Management of Data (SIGMOD'00)*, pp.82-92, 2000.

[18] B.-U. Pagel, H.-W. Six, H. Toben, P. Widmayer: "Towards an Analysis of Range Query Performance in Spatial Data Structures". *Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS'93)*, pp.214-221, 1993.

[19] T. Reinartz: "Similarity-Driven Sampling for Data Mining". *Proc. European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)*, pp.423-431, 1998.

[20] Y. Theodoridis, T. Sellis: "A Model for the Prediction of R-tree Performance". *Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS'96)*, pp.161-171, 1996.

[21] H. Toivonen: "Sampling Large Databases for Association Rules". *Proc. Int. Conf. on Very Large Databases (VLDB'96)*, pp.134-145, 1996.

[22] M. Vassilakopoulos, Y. Manolopoulos: "On Sampling Regional Data". *Data Knowledge Engineering (DKE)*, Vol. 22 No. 3, pp. 309-318, 1997.

[23] J. S. Vitter: "Random Sampling with a Reservoir". *ACM Transactions on Mathematical Software*, Vol.11, No.1, 37-57, 1985.

[24] M. Zaki, S. Parthasarathy, W. Li, M. Ogihara: "Evaluation of Sampling for Data Mining of Association Rules". *Proc. Workshop on Research Issues in Data Engineering (RIDE'97)*, 1997.

[25] S. Zhou, A. Zhou, J. Cao, J. Wen, Y. Fan, Y. Hu: "Combining Sampling Technique with DBSCAN Algorithm for Clustering Large Spatial Databases". *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'00)*, pp.169-172, 2000.