Available online at www.sciencedirect.com

SCIENCE DIRECT®

Data & Knowledge Engineering 57 (2006) 37–63

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

# Indexed-based density biased sampling for clustering applications ☆

Alexandros Nanopoulos [a], Yannis Theodoridis [b], Yannis Manolopoulos [a,*]

[a] *Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece*
[b] *Department of Informatics, University of Piraeus, Greece*

Received 14 September 2003; received in revised form 10 December 2004; accepted 24 March 2005
Available online 22 April 2005

**Abstract**

Density biased sampling (DBS) has been proposed to address the limitations of Uniform sampling, by producing the desired probability distribution in the sample. The ease of producing a random sample depends on the available mechanism for accessing the elements of the dataset. Existing DBS algorithms perform sampling over flat files. In this paper, we develop a new method that exploits spatial indexes and the local density information they preserve, to provide good quality of sampling result and fast access to elements of the dataset. With the proposed method accurate density estimations can be produced with respect to factors like skew, noise or dimensionality. Moreover, significant improvement in sampling time is attained. The performance of the proposed method is examined analytically and experimentally. The comparative results illustrate its superiority over existing methods.
© 2005 Elsevier B.V. All rights reserved.

*Keywords:* Sampling; Indexes; Density bias; Clustering; Data mining

☆ A preliminary version of this paper appeared in the proceedings of the CIKM'02 conference.
* Corresponding author. Tel.: +302310 991912; fax: +302310 991913.
*E-mail addresses:* alex@delab.csd.auth.gr (A. Nanopoulos), ytheod@unipi.gr (Y. Theodoridis), manolopo@delab.csd.auth.gr (Y. Manolopoulos).

0169-023X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2005.03.003

## 1. Introduction

The scalability of data analysis tasks to very large databases can be accomplished with data reduction schemes [3]. In data mining, sampling is a data reduction scheme that has found broad applications [16,13,28,31,25,17]. For the data mining task of clustering several algorithms capitalize on sampling as means of improving their efficiency [11,32,7,19]. They are based mainly on *Uniform random sampling*, with which every point has the same probability of being included in the sample.

Instead of Uniform sampling, the derivation of the desired probability distribution in the sample is more valuable for the task of clustering. An important example 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 sample. Density biased sampling (DBS) has been proposed in [22,15], 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. Although Uniform sampling can guarantee the inclusion of points from small clusters if very large samples are obtained, DBS accomplishes the same objective with much smaller samples (as proven in Theorem 1 of [15]). Consequently, DBS can be significantly more effective and efficient than Uniform sampling. It is interesting to notice that DBS may be considered as a generalization of Uniform sampling, since it can be reduced to the latter when the local density information is not taken into account [22].[1]

For instance, let two clusters depicted in Fig. 1. The rightmost consists of 50,000 points and the leftmost of 1000. The density (i.e., the number of points in each cluster with respect to its area) of the small cluster is about two times higher than that of the larger cluster.[2] By taking a Uniform sample of 1%, that is 510 points, the result of the random selection is that only nine 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 1000/51,000. 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.

### 1.1. Motivation

Existing DBS algorithms [22,15] use flat files, by performing one or more database scans and developing estimators for the approximation of local density. According to [3], the ease of producing a random sample depends on the available *sampling frame*, that is, the mechanism for accessing the elements of the dataset. Spatial Database Systems exploit access methods for efficient data organization. Sampling from spatial indexes has been introduced by Olken and Rotem [21]. How-

[1] DBS can also be considered as a specialization of *stratified* sampling, which partitions the data into mutually disjoint "strata" and then obtains a uniform random sample from each stratum.

[2] Small clusters may have the same density as larger ones, but higher from those parts of the space that do not contain clusters.

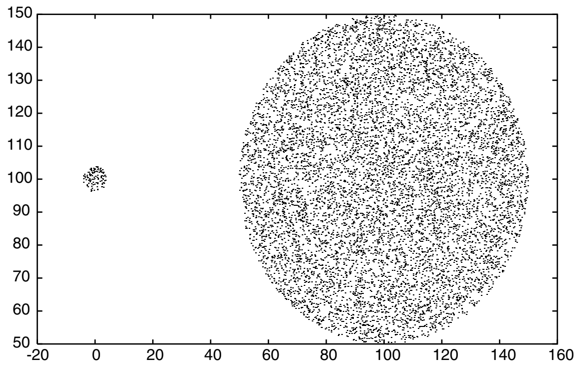

Fig. 1. An example of two clusters with different number of points.

ever, they focused on Uniform sampling, which does not consider density-bias. Spatial indexes (like the R-tree [10] and variants) achieve a clustering of objects within their nodes, which preserves the local density information and can provide good density approximations. Moreover, the index can comprise means for efficiently accessing the examined points. Conclusively, the exploitation of the existence of spatial indexes can result to an effective and efficient frame for density biased sampling, in terms of quality of sampling result and execution time for the sampling procedure.

### *1.2. Contribution*

In this work, we are interested in answering DBS queries with means of spatial indexes. In a way analogous to other spatial predicates, we focus on developing methods that exploit the properties of existing indexes to advocate the generation of the sample (for instance, in an analogous way [26] exploits the proximity that is preserved by existing indexes for the finding of nearest-neighbors). For comparison purposes, we initially consider extensions of the approach of [21,20], for the case of DBS. Then, we describe a new algorithm, which is based on a different approach and is able to present significant improvements in terms of efficiency. The proposed approach 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 small 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 to avoid the overhead of existing methods in terms of sampling time.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 contains the problem description and formulates the criteria that an algorithm for DBS from spatial indexes should fulfill. In Section 4, for comparison purposes, we examine the adaption of

Acceptance/Rejection algorithms, which are based on the paradigm of [20,21]. In Section 5, we present the proposed algorithm, which is based on a novel approach. Also, we formally show that it maintains the criteria required by DBS. Section 6 describes the analysis of the cost of the developed algorithms (the adaptations and the novel one), whereas Section 7 contains the experimental results both on sampling quality and efficiency, which illustrate the superiority of the proposed method. Finally, Section 8 concludes this work.

## 2. Related work

### 2.1. Non-uniform sampling for clusters with skew sizes

Palmer and Faloutsos [22] introduced the problem of non-uniform sampling for clusters with skew sizes, by over-sampling smaller clusters and under-sampling larger ones. The probability that a given point $x$, of a cluster $c$, is included in the sample is $P(\text{include point } x | x \in c) = f(n_c)$, where $n_c$ is the number of points in $c$. If $s$ is the expected sample size and $G$ is the total number of clusters, then [22]:

$$f(n_c) = \frac{s}{n_c^e \sum_{i=1}^{G} n_i^{1-e}}, \quad 0 \leqslant e \leqslant 1 \tag{1}$$

Since clusters are not known apriori, [22] 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 according to the probability given by Eq. (1), using the derived groups instead of the unknown clusters. Since this probability is inversely proportional to $n_g$, points from small groups have higher probability of being included (the procedure can be tuned by the value of $e$). In [22] 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 [22] 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 [15] and also verified by results in our work).

### 2.2. Density biased sampling from flat files

Kollios et al. [15] 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 [15] 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:

$$P(\text{include point } x) = \frac{s}{k} f^a(x) \tag{2}$$

In Eq. (2), $s$ 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 [15] show that the above method achieves improved sampling quality compared to [22] and Uniform sampling. However, the computation of the density estimator presents a significant time overhead, since it requires several database scans and high CPU cost. Evidently, in case where no updates occur in the data set, density estimation (for each point) can be computed once and, thereafter, this pre-computed information can be used each time DBS is applied. However, this approach still requires (during DBS) the examination of each point, which (as will be shown) leads to high cost due to CPU and I/O overhead.

In contrast, with the proposed approach, only a small fraction of points is examined during DBS (leading to smaller I/O cost), and the density information is kept on a per-node basis, i.e., not for each point (leading to smaller CPU cost). In case of dynamic updates in the data set, [15] requires a backing sample that will track changes and will identify when the density estimator should be modified. Although [15] does not address this issue, one can follow the lines of existing approaches, e.g., [9], in order to avoid the re-computation of density estimator after each update and to improve the average update times.

### 2.3. Uniform sampling from tree indexes

Olken and Rotem [21] have introduced the problem of Uniform sampling from spatial indexes. The ranked-tree algorithms, e.g., the Partial Area R-tree [21] or the pseudo-ranked extension of [2], are based on the iterative location of the $k$th entry in the tree, for a random $k$ at each iteration. These algorithms were proposed for Uniform sampling. For the purposes of DBS they can be utilized, if they use the ranking information to select a leaf at random, and then to select the sample points in an acceptance/rejection step. However, this may contrast the objective for which the ranking algorithms were proposed, i.e., to avoid the use of acceptance/rejection altogether. The algorithms that are based on acceptance/rejection (A/R) [20] were also proposed for Uniform sampling. However, they can be extended to the case of DBS.

For comparison purposes, such extensions are considered herein as base-line algorithms. Nevertheless, since they take into account all tree levels, their execution time is impacted. As will be described, the proposed algorithm differentiates from the A/R algorithms in the following ways: (1) first, it proposes the exploitation of spatial indexes for DBS and not for Uniform sampling (which was the focus of previous work). (2) It avoids the examination of all tree levels (in contrast to extended iterative and batched A/R algorithms); instead, condensed information is maintained about the leaf nodes, which helps to avoid traversing the tree in a root-to-leaves manner. (3) The described optimizations for the proposed algorithm help in avoiding the examination of a large number of leaves (compared to extended iterative and batched A/R algorithms). (4) The expected sample size can be guaranteed (in contrast to extended batched A/R algorithms). All these differences help in significantly reducing the execution time of DBS.

Other related work on Uniform sampling from spatial indexes includes [18,29]. Finally, [8] 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, [8] 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 [8] where it is determined by the number of leaf nodes.

## 3. DBS with spatial indexes

In spatial indexes, the update operations provide a clustering of points within their nodes so as to maximize selectivity during query processing [23,27]. In the sequel we focus on R-tree [10] 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 and to minimize dead space. This kind of clustering achieved by the R-tree preserves the local density information within tree nodes, since areas that are in proximity tend to belong to the same cluster, thus to have similar density.

For instance, Fig. 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 Fig. 2a nodes are represented by their minimum bounding rectangles (MBRs)[3]). 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 Fig. 2b. Evidently, in this case no clustering is achieved within nodes and the local density is not preserved.

Of course, the R-tree by itself cannot result in perfect clustering, due to its inherent restrictions (basically the lower/upper limit in the node size and the splitting of a node that overflows into always two nodes—see [6] for more details on this issue). Nevertheless, for the purpose of DBS we are interested in the approximation of local density, which as will be shown in the following, will be effective enough.

The problem of exploiting the organization of points within the R-tree nodes for purposes of DBS presents some analogies to the selection of representative points from the leaves of the R-tree [8]. Although the objectives of these two cases are quite different (see Section 2), the common motivation is to use the clustering information that is stored in the R-tree in order to advocate a (generic) clustering algorithm. According to the description in [8], the limitations in the R-tree

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

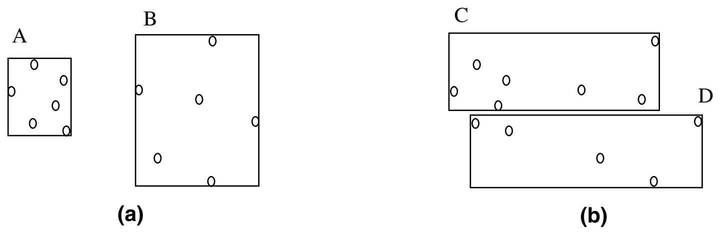

Fig. 2. An example of storing two clusters with different densities.

(that were also described previously) affect the effectiveness of an approach that will base the clustering of points directly on the R-tree, because clusters are distributed among several leaves which, moreover, may not be siblings (this affects efficiency, since the examination of a large number of leaf combinations may be required). Therefore, although one may consider an approach that will merge/split the leaves so as to cluster the points, such an approach has not been followed so far. Also, its objectives are different than the ones examined by the problem in this paper, that is, the exploitation of the R-tree so as to advocate clustering algorithms through DBS.

Let $L$ denote the set of leaf nodes of the R-tree. As described in the following (c.f., first criterion of Criterion 1), it is desired that points belonging to the same leaf node to have identical probability of being included in the sample. For this reason we assign the same local density to all points in a leaf. Let $f_j$ denote the fanout of a leaf node $j$.[4] 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-dimensional 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

$$D_i = \frac{f_i}{\text{Volume}(\text{MBR}_i)}, \quad i \in L \tag{3}$$

The inclusion of points into the sample has to be done according to the density function of Eq. (3). Palmer and Faloutsos [22] describe four criteria that a point inclusion probability function should satisfy for DBS, given a partitioning of points into specified groups. Kollios et al. [15] give one additional criterion, that the proposed method in [15] preserves the relative densities. Since the grouping of points into clusters is not known apriori, Palmer and Faloutsos [22] consider the corresponding criteria for the approximate grouping obtained by using the proposed hash table [22].[5] For the case of DBS from an R-tree, we consider the grouping of points into the set of leaf nodes $L$. Therefore, we can state Criterion 1, which gives the criteria that a DBS algorithm should satisfy. Notice that in the first part of the second criterion, the notion of *density preserving sampling* is according to the definition given [22]. Also, the second part of the second criterion uses the notion of *relative density* according to the definition of [15]. The remaining criteria are based on the description of [22].

[4] For simplicity, we assume that the internal and leaf nodes have the same maximum fanout (i.e., capacity).

[5] Although this is not given explicitly in [22], the corresponding criteria are used to derive the proposed approximate algorithm in [22], therefore they hold for the given approximate grouping.

Table 1
Symbol table

| Symbol | Definition |
|---|---|
| $d_i$ | Normalized density of leaf $i$ |
| $f_i$ | Fanout of node $i$ |
| $f_{\max}$ | Max fanout (internals and leaves) |
| $N$ | Number of all data points |
| $L$ | Set of all leaf nodes |
| $h$ | R-tree height |
| $a$ | Tuning parameter for DBS |
| $f_{\text{avg}}$ | Average value of $f_i$ (internals and leaves) |
| $d_{\text{avg}}$ | Average value of $d_i$ |
| $\beta$ | $f_{\text{avg}}/f_{\max}$ |
| $\beta_0$ | $f_0/f_{\max}$ ($f_0$ the root fanout) |
| $s$ | Sample size |

**Criterion 1.** Assuming a partitioning of points into specified groups, the probability function for including points in the sample has to satisfy the following criteria:

1. Within a group, points are selected uniformly.
2a. The sample is density preserving.
2b. The relative densities in the sample are preserved when $a > 0$ ($a$ is the tuning parameter). That is, if a region of the dataset has higher density than another, then the same is expected in the sample.
3. The sample is biased by the group densities.
4. The expected sample size is $M$.

The first criterion states that points from the same cluster should be handled equivalently. The second and third ones result from the density-bias requirement. Finally, the fourth states that size of the random sample should be a user-defined value. Table 1 contains the definition of symbols used henceforth.

## 4. A/R algorithms

To perform DBS from an R-tree, one could form a random path from root to a leaf $i$. The path can be formed by visiting nodes and selecting, each time, a child-pointer to follow at random. At $i$, according to density $D_i$, it is decided whether to include in the sample a point from $i$. This procedure can be repeated until the required sample size is obtained. However, the inclusion probability of a point would incorrectly depend not only on $D_i$ (as required by Criterion 1), but also on the path from root to leaf $i$. Evidently, a point from a leaf, for which the corresponding path consists of nodes with small fanouts, is more probable to be selected than from another with larger fanouts. Olken and Rotem [20,21] describe A/R algorithms to avoid the above problem in the case of Uniform sampling. However, these approaches do not consider the density bias. Therefore, for comparison purposes, we initially present extensions of A/R algorithms, which address the problem of DBS from an R-tree. In the following section we describe a new DBS algorithm.

## 4.1. Iterative algorithm

Iterative A/R sampling algorithms [20] descend the tree through root-to-leaf paths, exercising a random choice of a child-pointer at each node in the path. The differences in the nodes' fanouts are compensated by performing the selection as if each node had the maximum fanout $f_{\max}$. In the case of DBS, 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 probability[6]), and let $d_i$ denote $D_i/D_{\max}$. Fig. 3 describes an A/R DBS algorithm, which is based on the corresponding *Early-Abort Iterative* algorithm (EAI) of [20]. EAI uses a tuning parameter, denoted as $a$. Parameter $a$ controls the rate of DBS at different cases, and its tuning is described in Section 7.1.

As an example, consider the points illustrated in Fig. 4a, which are organized in three MBRs: $A$, $B$, and $C$. The corresponding R-tree is depicted in Fig. 4b, where it is assumed that $f_{\max} = 4$. Let the area of $A$ be equal to 4 units, of $B$ equal to 2 units, and of $C$ equal to 6 units. The corresponding densities are: $D_A = 1$, $D_B = 1.5$, $D_C = 0.3$. Therefore, the normalized densities are: $d_A = 0.6$, $d_B = 1$, $d_C = 0.2$. Also, let $a$ be equal to 1. For the selection of a point, EAI will start from the root. Since the root node is not a leaf, a child $j$ is selected at random. Assume that the child $A$ is selected. A random number generation is performed for the $r$ variable. Let $r$ become equal to 0.5. Since $f_A/f_{\max} = 4/4 = 1 > r$ and the current node is a leaf, another random number generation is performed for the $r$ variable. Assume that $r$ now becomes equal to 0.3. Since $d_A^a = 0.6 > r$, a point is selected at random from leaf $A$. Let that point $a_2$ is selected. If more than one points are required for the sample, then another iteration of EAI will have to start.

Since at each iteration a point is included according to a probability, more than $M$ iterations may be required to obtain a sample of $M$ points. The probability of a single point to be included in the sample during each iteration is given by the following lemma.

**Lemma 1.** *Let $f_{\max}$ be the maximum fanout, $f_0$ the root fanout and $h$ the height of the R-tree. For each iteration of EAI, the probability that a point $x$ is included from a leaf i, is*

$$P(\text{include a point } x \text{ from } i) = f_0^{-1} \cdot f_{\max}^{-h+1} \cdot d_i^a. \tag{4}$$

**Proof.** Based on [20], let $P_i$ denote the path from root to leaf $i$, and also let $r$ be the root node. At each node $j \in P_i$ ($j \neq r$), $P_i$ is *not* aborted with probability $f_j/f_{\max}$ (at step EA2). Therefore, the complete $P_i$ (up to leaf $i$) is accepted with probability $\prod_{j \in P_i, j \neq r} f_j/f_{\max}$. At each $j \in P_i$, the next node of $P_i$ is selected with probability $1/f_j$, therefore a point in $i$ is reached with probability $1/f_0 \cdot \prod_{j \in P_i, j \neq r} 1/f_j$. Additionally, at leaf $i$, acceptance is also decided (at step EA4) with probability $d_i^a$. Hence, the overall probability of including a point of $i$ in the sample, during each iteration, is the product of these partial probabilities, i.e., $\frac{1}{f_0} \cdot \prod_{j \in P_i, j \neq r} (\frac{1}{f_j} \cdot \frac{f_j}{f_{\max}}) \cdot d_i^a = f_0^{-1} \cdot f_{\max}^{-h+1} \cdot d_i^a$. □

[6] $D_{\max}$ 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 value.

```
EA1.  Start with $j$ = root and execute EA3.
EA2.  Generate a random number $r \sim U(0,1)$.
      If $r > f_j/f_{max}$, then execute EA1.  /* abort */
EA3.  If $j$ is a leaf, then execute EA4.
      Else, assign to $j$ a child at random, execute EA2.
EA4.  Generate a random number $r \sim U(0,1)$.
      If $r < d^a_j$, then select a point at random.
EA5.  If $M$ points have been selected, then terminate.
      Else, execute EA1.
```

Fig. 3. The Early-Abort Iterative DBS algorithm.

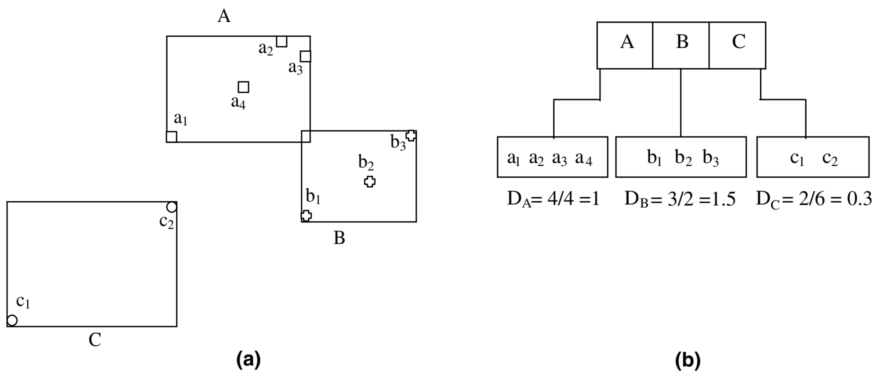

Fig. 4. Example of points organized in three MBRs (a) and of the corresponding R-tree (b).

Therefore, as required, the probability depends only on the density value and not on the fan-outs of the root-to-leaf paths. Based on Lemma 1, EAI is in accordance with Criterion 1.

**Proposition 2.** *The probability function given by Eq.* (4), *for including points with the EAI algorithm at each iteration*, *satisfies Criterion* 1.

**Proof.** We separately prove each of the four criteria of Criterion 1.

1. The probability of including a point from each $i \in L$, is $P(\text{include point } x|x \text{ in leaf } i) = f_0 \cdot f_{\max}^{-h+1} \cdot d^a_i$ (Eq. (4)). Therefore, the inclusion probability at each iteration is the same for all points of $i$. Hence, points are selected with uniform probability from $i$.

2a. Let that $\{x_1, \ldots, x_{f_i}\}$ denote the $f_i$ points in a leaf $i$. According to [22], we have to show that, for each iteration, the expected sum of the weights of the sample points from each leaf $i$ is proportional to $f_i$, i.e., $\sum_{j=1}^{f_i} w_j \cdot P(x_j) = \kappa \cdot f_i$, where $P(x_j) = P(\text{include point } x_j|x_j \text{ in leaf } i)$, given by Eq. (4). By setting $w_j = 1/P(x_j)$ (i.e., all points of $i$ are assigned the same weight, due to case 1), we have $\sum_{j=1}^{f_i} 1/P(x_j) \cdot P(x_j) = f_i$. Therefore, the required equality holds for $\kappa = 1$.

2b. Let $i$ and $j$ be two leaves. Without loss of generality, we assume that Volume(MBR$_i$) = Volume(MBR$_j$) (see the identical assumption made in proof of Lemma 1 in [15]). If $d_i \geqslant d_j$, then we have to show that more points are included in the sample from $i$ than from $j$. Let that $T$ iterations of the EAI algorithm are required to obtain the sample. At each iteration the probability of including a point from $i$ in the sample is given by Eq. (4). Thus, the number of points that will be selected from $i$ is a random number that follows, Binomial distribution and for its expected value, $S_i$, for $T$ iterations it holds that $S_i = T \cdot f_0 \cdot f_{\max}^{-h+1} \cdot d_i^a$. Similar for leaf $j$, $S_j = T \cdot f_0 \cdot f_{\max}^{-h+1} \cdot d_j^a$. Hence, $S_i \geqslant S_j$ when $a > 0$. Therefore, in the sample, the expected density of the region corresponding to $i$ is larger than that of $j$.

3. Since $P(\text{include point } x | x \text{ in leaf } i) \propto d_i^a$, for each leaf $i$ the inclusion of points, at every iteration, is biased from its density. It has to be noticed that when $a = 0$, the probability degenerates to uniform and the EAI algorithm to Uniform sampling.
4. The EAI algorithm performs as many iterations, until exactly $M$ points are included in the sample. □

### 4.2. Batch algorithm

The batch A/R sampling algorithms that are described in [20], have the objective of reducing the rereading of nodes incurred by iterative algorithms. Herein, we describe the batch analog of the Early-Abort Iterative algorithm, which is denoted as *Batch Early-Abort* (BEA). BEA performs a depth-first search of the tree and allocates the incoming sample size to each node (initially, it starts from the root node with a gross sample size $s'$). At each node $i$ (besides the root), BEA performs the A/R criterion (similar to EAI) by generating a binomial random variable $x_i \sim B(s_i, f_i/f_{\max})$, where $s_i$ is the sample size allocated to $i$. The allocation of the sample size to the children nodes is done by means of a multinomial random vector (more details can be found in [20]). The extension of BEA to the case of DBS is done in a way analogous to the extension of EAI. Thus, if $s_j$ is the sample size initially allocated to leaf $j$, the net sample size is $x_j$, where $x_j \sim B(s_j, f_i/f_{\max} \cdot d_j^a)$.

It is easy to show (in a way analogous to EAI) that BEA satisfies the first three criteria of Criterion 1. For the fourth criterion, however, BEA returns a sample size of random size [20]. If it is much smaller than the expected sample size, then as described in [20], BEA may have to be repeated until enough points have been included in the sample.

## 5. Selective pass algorithm

In this section, we present a new method for DBS from an R-tree, called *Selective Pass* (SP), which follows a different approach than A/R algorithms. SP aims at improved efficiency while maintaining the fulfilment of the four criteria of Criterion 1. Initially we give a basic version of SP and then we describe the optimized form of the algorithm.

Let $N$ be the total number of points in the tree. A/R algorithms consider all the levels of the tree and use the A/R criterion so as to compensate the differences in the fanouts. In contrast, SP considers directly the partitioning of the $N$ points into the set $L$ of leaf nodes. It performs an

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

SP1. Start from the left-most leaf, $i$.
SP2. Generate a random number $s_i \sim B(t, \frac{f_i}{N} \cdot d_i^a)$.
SP3. Select at random $s_i$ points from $i$
     (selection with replacement is performed).
SP4. If all leaves have been examined, then terminate.
     Else, set $i$ to the next leaf and execute SP2.
```

Fig. 5. The basic version of the Selective Pass DBS algorithm.

examination of the members of $L$. For a $i \in L$, SP decides whether to include a point from $i$ into the sample according to $d_i$, and selects a point at random. However, to avoid the rereading of leaf nodes, SP assigns to each leaf a number $t$ of trials ($t$ is the same for each leaf). The basic version of the algorithm is depicted in Fig. 5 ($a$ is the parameter that controls the degree of bias, and its tuning is described in Section 7.1).

For the example of the R-tree in Fig. 4b, SP will start from the left-most leaf, $A$. Let that the generated value $s_A$ will be equal to 2. Thus, two points will be selected, e.g., $a_2$ and $a_4$. Nest, SP will examine leaf $B$. Let that the generated value will be $s_B = 1$, and that point $b_2$ is selected. Finally, at leaf $C$, assume that $s_C = 1$ and that point $c_1$ will be selected.

In SP, the leaf nodes of the R-tree are examined sequentially (where at each leaf, the link to the next leaf is available[7]). Step SP2 corresponds to a series of $t$ Bernoulli trials for the inclusion of points from leaf $i$ into the sample, where the probability of success for each trial is equal to $f_i/N \cdot d_i^a$. Consequently, the total number $s_i$ of successes (i.e., the number of included points from leaf $i$) is a random variable that follows binomial distribution. At step SP3, $s_i$ out of $f_i$ points from $i$ are selected. Each point is selected with uniform probability $1/f_i$. Selection with replacement is performed to maintain the latter probability equal to $1/f_i$. Otherwise, this probability will be different after each selection (an analogous reasoning have been followed in [20]). Thus, for the point-selection probability of SP it holds that:

**Lemma 3.** *For each trial of SP, the probability to include a point x from a leaf i, is*

$$P(\text{include a point } x \text{ from } i) = N^{-1} \cdot d_i^a. \tag{5}$$

**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, $P(\text{include point } x | x \text{ in leaf } i) = f_i/N \cdot d_i^a \cdot 1/f_i = N^{-1} \cdot d_i^a$. □

Consequently, it is easy to show, in a way similar to Proposition 2 for the EAI algorithm, that the probability function of Eq. (5) satisfies the first three criteria of Criterion 1 (by substitution of the $f_0 f_{\max}^{-h+1}$ factor with the $N^{-1}$). For the fourth criterion, we give the following proposition.

[7] Since in commercial RDBMS R-tree is implemented on top of $B^+$-trees, this linkage is typically available [24].

**Proposition 4.** *If* $t = M\left(\sum_{j\in L} f_j/N \cdot d_j^a\right)^{-1}$ *is the number of trials assigned to each leaf node, then the probability function of Eq. (5) satisfies the fourth criterion of Criterion* 1 (*i.e., the expected sample size is equal to M*).

**Proof.** From each leaf $j$, $s_j$ points are included in the sample, where $s_j \sim B\left(t, f_j/N \cdot d_j^a\right)$. 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, $E(s) = E\left(\sum_{j\in L} s_j\right) = \sum_{j\in L} E(s_j)$. By letting $E(s) = M$ and substituting the value of $E(s_j)$, we get $M = \sum_{j\in L}\left(t \cdot f_j/N \cdot d_j^a\right)$. Solving for $t$, the required equality follows. □

It has to be noticed that SP could be simplified by selecting exactly $t \cdot f_j/N \cdot d_j^a$ (the expected value) points from each leaf $j$. However, this number is non-zero for each $j$ and, even in the cases it is small (e.g., 1), SP would still have to fetch $j$. When $s_i$ is random, then for a possibly significant number of leaves this number can be equal to zero (see Fig. 7b), and thus SP can entirely avoid to fetch these leaves.

To verify the aforementioned argument, we use synthetic data sets (which are detailed in Section 7) and measured the percentage difference (denoted as sample size error) between the requested ($M$) and resulting sample size (we took the absolute values between the differences in sizes, because the resulting sample size can be larger or smaller than the requested one). The results are illustrated in Table 2 with respect to requested sample size (given in percentage of the data set size).

## 5.1. Optimized SP

Additionally to the basic version, an optimization can be developed for SP. It maintains for each leaf $i$, its address, the $d_i$ and $f_i$ values. Let this information about leaf nodes be denoted

```
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 elements of $I_L$ have been examined, then terminate.
      Else, get the next entry of $I_L$ and execute SP2.
```

Fig. 6. The optimized version of the Selective Pass DBS algorithm.

Table 2
Sample size error w.r.t. $M$ value (%)

| Sample size error (%) | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 1.41 | 1.85 | 0.89 | 1.66 | 0.85 |

as $I_L$. Then, a leaf $i$ has to be fetched from secondary storage only when $s_i > 0$, at step SP2 of the basic version. Thus, the reading of nodes that will not contribute to the sample is avoided and a reduction in the I/O overhead is attained. The optimized version of SP is depicted in Fig. 6.

For fair comparison with the other algorithms, the entries of $I_L$ are read from disk during the execution of DBS (each time, only the current page of $I_L$ entries have to remain in main memory). 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 due to the avoidance of reading several leaf nodes (as it is shown in Section 6). It can be easily shown that for, e.g., five dimensional data, $I_L$ occupies only 15% of the space required by the upper levels of the index (i.e., excluding the leaves).

Evidently, for static data the entries of $I_L$ can be obtained during the index creation. For dynamic data, the entries of $I_L$ can be easily accommodated in main memory, considering the large memory sizes today. It is straightforward to maintain the entries of $I_L$ during index updates. After an update in a leaf $i$ (insertion/deletion of points), the calculation of the new values of $f_i$ 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 $\mathrm{MBR}_i$, which are also available during the update operation at $i$. Therefore, SP can efficiently handle dynamic updates.

## 6. Analytical comparison

To analytical compare the cost of the described R-tree based DBS algorithms (EAI, BEA and SP), we consider the cost measure to be the number of referenced R-tree nodes. In the sequel, $C_{\mathrm{EAI}}$, $C_{\mathrm{BEA}}$ and $C_{\mathrm{SP}}$ denote the corresponding costs. Let $\beta$ denote the $f_{\mathrm{avg}}/f_{\mathrm{max}}$ and $s$ the required sample size. To derive less complicated results (i.e., more compact formulae), we are based on the aforementioned average statistics. For the same reason, similarly to [20], we do not take into account the caching of nodes besides the root.

For EAI, based on [20] it can be shown that

$$C_{\mathrm{EAI}} = \frac{\beta^{h-1} - 1}{\beta - 1} \cdot s \cdot f_{\mathrm{avg}}^{-h+1} \cdot d_{\mathrm{avg}}^{-a} \tag{6}$$

BEA commences the depth-first search with a gross sample size $s'$. Following the approach of [20], it can be easily shown (by considering the probability of success at each trial of BEA) that $s'$ should be set to $s \cdot \beta^{-h+1} \cdot d_{\mathrm{avg}}^{a}$. Therefore, by considering for each level separately the Cardenas' function $Y(k,m)$ [20], it holds that

$$C_{\mathrm{BEA}} = 1 + \sum_{j=1}^{h-1} Y\left(s' \cdot \beta^{j-1}, f_0 f_{\mathrm{avg}}^{j-1}\right) \tag{7}$$

where $Y(k,m) = m(1 - (1 - (1/m))^k)$.

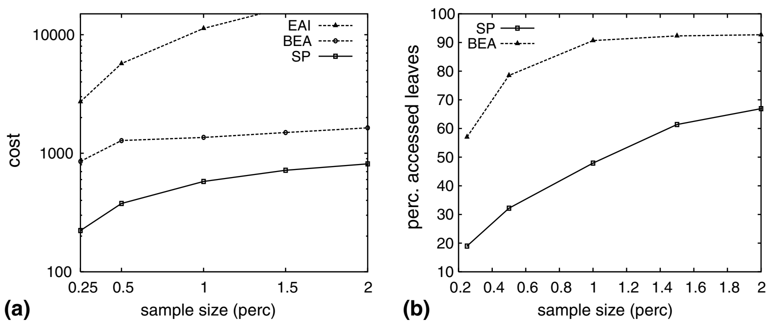

Fig. 7. Analytical comparison of R-tree based DBS algorithms.

For SP (optimized version), the expected number of non-referenced leaves is equal to $\sum_{i \in L}(1 - \frac{f_i}{N} \cdot d_i^a)^t$. By considering the average fanout and density, and by taking into account the reading of $I_L$, we get

$$C_{\text{SP}} = |L|\left[1 - \left(1 - \frac{f_{\text{avg}}}{N} \cdot d_{\text{avg}}^a\right)^t\right] + O(|L|/B) \tag{8}$$

The estimation accuracy of the developed cost measures has been examined and found to be around 3% for $C_{\text{SP}}$ and $C_{\text{EAI}}$, and around 12% for $C_{\text{BEA}}$. Therefore, using these cost measures, we focus on the impact of sample size. We used the North East data set and the experimental setting described in Section 7.1. Fig. 7a illustrates the results.

As expected, the cost of EAI is linear in the sample size. Due to its high *rejection rate* [2], EAI is clearly outperformed by the other two algorithms. SP performs better than BEA (by a factor of two, for smaller sample sizes), since it avoids the examination of upper-levels. Focusing on SP and BEA (EAI is omitted due to its large cost), we also present the cost as a percentage of the accessed leaves (w.r.t. the total number of leaves). The results are given in Fig. 7b. As shown, SP requires the access to a small percentage of leaves, compared to BEA. The experimental results in the following section will verify the aforementioned conclusions.

## 7. Performance evaluation

To evaluate the performance of the proposed method, we conducted a series of experiments using synthetic and real datasets. For comparison, we examine the algorithms of [15] and [22], which are referred as Biased Sampling (BS) and Grid Biased Sampling (GBS) respectively (according to the notation in [15]). We also examine Uniform sampling (US), following the algorithm of [30].

We consider both effectiveness and efficiency. Effectiveness is examined by measuring the quality of sampling with respect to clustering result, along the lines of [15,22]. 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. (BEA produces similar results to EAI, since they have the same inclusion probability function; thus for brevity, we omit the results for BEA on sampling quality and we only report on its efficiency.)

## *7.1. Experimental configuration*

We implemented SP, EAI and BEA algorithms in C. The code for BS was provided by the authors of [15] and the code for GBS is available.[8] 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. Evidently, the clustering result depends both on the selected clustering algorithm and the parameter setting that most of such algorithms use. Herein, we use the CURE clustering algorithm [11]. The reason is that we want to perform a fair comparison with the results in [15], which are also based on CURE (the parameters had the default setting described in [11], i.e., 10 representative points and 0.3 shrinking factor). However, since the clustering result depends on the used clustering algorithm, we also examine the case of the $C^2P$ algorithm [19]. The characteristics of this algorithm are that it is also based on sample data and it uses spatial indexes for the clustering procedure. Henceforth, for convenience, we denote that the $x$ sampling method finds a specified clustering result, by meaning that the clustering result is found using the clustering algorithm on the sample produced by the $x$ sampling method.

We examine real datasets and, in order to better control the data characteristics, we also 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. For this type of datasets, since we know the cluster centers, 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 (default value: 0.01). Due to space constraints, more details can be found in [22].

As described, all presented algorithms have a tuning parameter to determine the degree of bias in sampling. We tune the $a$ parameter of BS according to the indications in [15]. 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. Also, experimental results in [15] indicate that in the case of clusters with various densities where, additionally, the dataset contains noise, $a$ is set to $-0.25$ (i.e., the half of $-0.5$ that is used in case of no noise), so as to produce improved results. For SP, EAI and BEA we adopt the same setting as in [15] for the $a$ parameter. For GBS we set $e$ to 0.5 (the corresponding tuning parameter—see Eq. (1)), which corresponds to the IRBS method [22], because experimental results in [22] show that it gives the best performance.

Finally, the default number of kernels for BS was set to 1000 [15]. The default available memory size for the hash table of GBS [22] and for the buffer space used by EAI and BEA was set to the 20% of the dataset size.[9] We used the R*-tree [4] variant and the default page size was set to 4K. There-

[8] At http://www.cs.cmu.edu/People/crpalmer/dbs-current.tar.gz.

[9] For the case of pre-computed density information, SP is not affected by the buffer size, since $I_L$ is read from disk and in each run of the algorithms we flush the buffers (so as to isolate their results). The same applies for BS. Therefore, we used a relatively large buffer size to favor EAI and BEA in terms of efficiency (they may visit the same nodes more than once) and GBS in terms of effectiveness (a larger hash table allows for better estimation). For the case of not pre-computed density information, we separately examine the impact of buffering in Section 7.3.

fore, the maximum fanout (called henceforth fanout) is a function of dimensionality (i.e., it is not constant). Each measurement (both on sampling quality and efficiency) is the average of 5 runs.

## 7.2. Results on sampling quality

### 7.2.1. The NE dataset

We tested the examined algorithms using real datasets. We report results on the *North East* dataset (Fig. 8a), that was used in [15]. It contains 130,000 postal addresses (as two dimensional

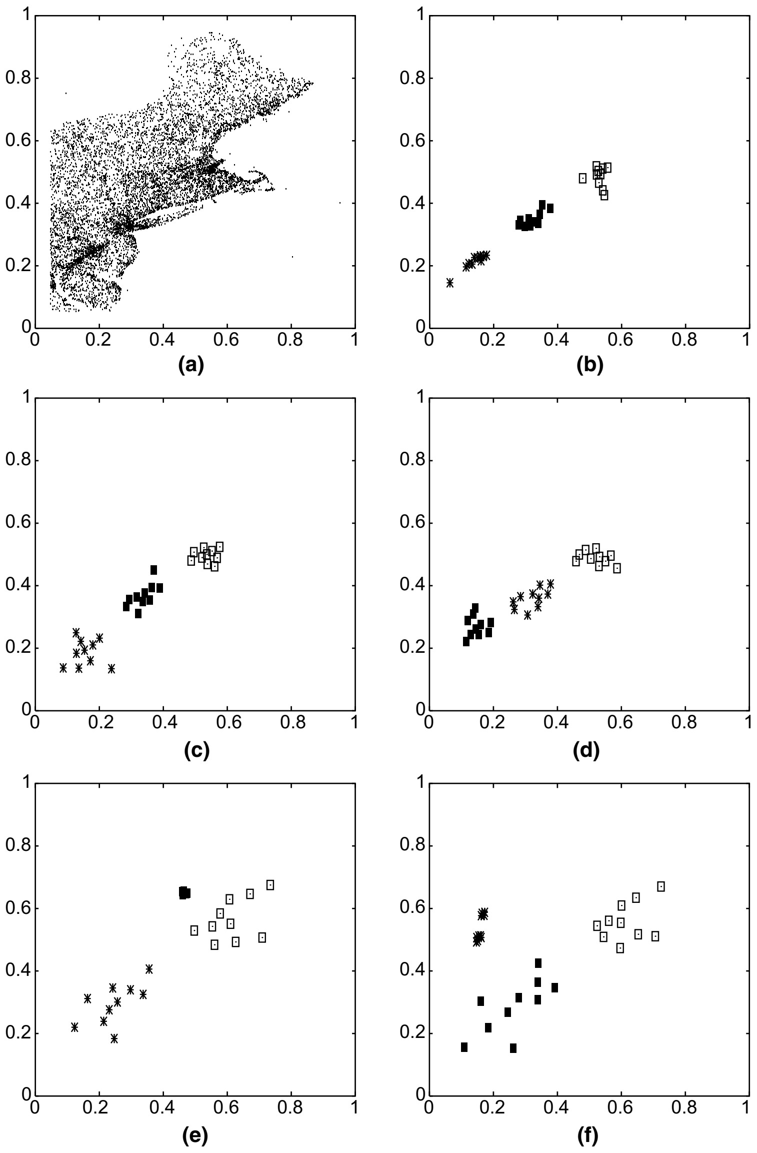

Fig. 8. (a) The North East dataset. (b)–(f) Results (through representative points EAI, SP, BS, GBS, US, repectively).

points), which represent three metropolitan areas: New York, Philadelphia and Boston. Also, there is a lot of noise in the form of uniformly distributed rural areas and smaller population centers. We used a sample of 1000 points and the CURE clustering algorithm (analogous results were obtained for $C^2P$, which are omitted for brevity). Fig. 8b–f illustrate the clustering results, through the representative points of each found cluster, for EAI, SP, BS, GBS and US respectively.

With SP, EAI, and BS the correct clustering result is produced (the three clusters are detected). However, this does not hold for GBS and US. Noise and outliers impact the quality of both GBS (because it favors them) and US (because it does not discriminate between noise and cluster points). These results are in accordance with the ones in [15] for this dataset.

### *7.2.2. Clusters with skew sizes*

We measured *NC* (the number of correctly found clusters) for the case of datasets which contain clusters with skewed sizes. Since clusters with very small sizes are likely to be missed by Uniform sampling, with these experiments we tested the effectiveness of DBS methods. 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). We present a case which is analogous to the *One Big* dataset of [22]. One cluster contained 50,000 points and the remaining ones had 500 points (100 times less). However, differently from the case in [22], 10% noise was added. Also, the number of kernels for BS was set to 3000 (since the default value of 1000 produced worst results). We used both the CURE and the $C^2P$ clustering algorithms. The results with respect to the sample size (given as percentage of dataset size) are depicted in Table 3a (for CURE) and in Table 3b (for $C^2P$).

Regarding the results for CURE, BS managed to find the correct clusters earlier than US (for sample size 2%) and it found more correct clusters in the cases where both did not correctly found all clusters, i.e., when $NC < 9$. US on the other hand found the correct clusters for the 4% sample size. These are in accordance with the conclusions in [15], i.e., BS outperforms US, by finding the correct clusters with a smaller sample size. GBS did not produce correct results, because it is im-

Table 3
*NC* w.r.t. sample size for the case of clusters with various sizes

| Algorithm | Sample size (%) | | | | |
|---|---|---|---|---|---|
| | 1 | 1.5 | 2 | 3 | 4 |
| *(a) Results for CURE* | | | | | |
| SP | 9 | 9 | 9 | 9 | 9 |
| EAI | 9 | 9 | 9 | 9 | 9 |
| BS | 6 | 8 | 9 | 9 | 9 |
| GBS | 2 | 3 | 6 | 7 | 7 |
| US | 4 | 5 | 7 | 8 | 9 |
| *(b) Results for $C^2P$* | | | | | |
| SP | 8 | 9 | 9 | 9 | 9 |
| EAI | 8 | 9 | 9 | 9 | 9 |
| BS | 6 | 8 | 9 | 9 | 9 |
| GBS | 2 | 3 | 7 | 7 | 7 |
| US | 3 | 5 | 8 | 8 | 9 |

pacted 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 and EAI perform better than the other methods, even at lower sample sizes, since they are not affected by skew in cluster sizes. Analogous results are produced for $C^2P$. However, for very low sample size (1%), SP and EAI miss one cluster. The reason is that the focus in the design of $C^2P$ was given on scalability to large input sizes. Therefore, the first phase of this algorithm was affected by the small sample size combined with the skew in the cluster sizes. Nevertheless, SP and EAI still perform better than the other methods.

### *7.2.3. Noise and dimensionality*

In this section we examine the impact of noise and dimensionality. Since CURE and $C^2P$ produced qualitative similar results and due to space constraints, we focus on the former.

Starting from the examination 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 4a, with respect to the amount of noise (given as a 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). As the percentage of noise increases, points from clusters are becoming less probable of being selected by US. On the other hand, as noise percentage increases, noisy points are becoming more probable to be selected by GBS, since it favors less dense groupings of points that noise tends to form.

SP and EAI clearly present the best performance. They correctly find all clusters in cases with low and medium noise (0–60%), whereas at very high noise percentage (80%) they start missing clusters. BS, is affected at a less high value, by start missing clusters at 60%. BS uses approximation with a kernel function, which is based on a uniform sample of $\beta$ points [15] and distributes the weights around each point from the uniform sample. As the noise percentage increases, the larger becomes the probability that noisy points to be included in the uniform sample used by the kernel

Table 4
*NC* w.r.t. (a) noise percentage, (b) dimensionality

| Algorithm | Noise percentage | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 10 | 20 | 30 | 40 | 60 | 80 |
| SP | 9 | 9 | 9 | 9 | 9 | 9 | 8 |
| EAI | 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 |

| | Dimensionality | | | | |
|---|---|---|---|---|---|
| | 5 | 8 | 10 | 12 | 15 |
| SP | 9 | 9 | 9 | 9 | 8 |
| EAI | 9 | 9 | 9 | 9 | 8 |
| BS | 9 | 9 | 8 | 8 | 8 |
| GBS | 7 | 6 | 5 | 5 | 5 |
| US | 7 | 6 | 5 | 4 | 4 |

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 and EAI, 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). Therefore, for medium noise percentage (e.g., 40–60%), SP and EAI maintain the quality of sample. The reason is that, for low and medium noise percentages, in those leaves that contain noise the larger fraction of points still belongs to clusters (i.e., not noisy points), whereas there exist a non-negligible number of leaves that do not contain noise. However, for very high noise percentage (80%), where most of the leaves contain many noisy points, both methods are affected.[10]

Next, we considered the impact of dimensionality.[11] Noise was set to 25% and the sample size was 2%. The results are depicted in Table 4b. 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 (since, as described in the previous experiments, US cannot effectively distinguish between noise and points in small clusters).

In contrast, SP finds the correct results in all cases with dimensionality equal or less than 12. For higher number of dimensions (e.g., 15) it starts missing clusters. The same applies for EAI. We note here that higher 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% 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.

## 7.3. 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 [15], we do not report results on their execution times.

The proposed approach mainly focuses on the exploitation of indexed datasets (i.e., for which a spatial index exists). This is analogous to the approach of [20] for Uniform sampling from datasets for which a $B^+$-tree index is already build.[12] Also, many previous approaches in processing spatial data exploit the proximity information that is already stored in the builded R-tree (for instance [26]). For non-indexed datasets, one can expect that the bulk-loading of the spatial index will

[10] To reduce the effect of noise, we can consider heuristics that, e.g., will determine the density of a leaf by the core of most central points (closest to the center of node). We address the examination of such heuristics as future work.

[11] It has to be noticed that, by using CURE or $C^2P$, we focus on full-dimensional clustering. For the particularities of very high dimensional spaces, specialized algorithms have been proposed [12].

[12] Otherwise, single-pass algorithms [30] 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.

dominate the total cost. It has to be noticed that the selection of a bulk-loading algorithm and its implementation details are orthogonal to the problem examined in our work. For the above reasons, our main results refer to the case of interest, i.e., of a builded index, thus they do not include the index creation time. Nevertheless, we also examine the viability of the proposed method for datasets for which an index does not exist. Finally, we note that in all experiments we include in our measurements the time to write the sample on disk.

### 7.3.1. R-tree based algorithms

First, we compare the developed R-tree based DBS algorithms. To give a clear comparison of the sampling time alone, we assume the existence of the index for all three methods. Fig. 9a illustrates the results with respect to the sample size (given as percentage) for 3-dimensional datasets with 100K points. Clearly, EAI presents the worst performance. SP outperforms BEA in all cases, where for larger sample sizes the performance difference is by a factor of two. These results are in accordance with the analytical comparison that is presented in Section 6, in terms of relative performance between the algorithms. Based on the above, to present a more clear comparison between SP and BEA, in the following we omit further results on EAI.

We now move on to examine the scalability to the dataset size. Fig. 9b presents the results with respect to the number of points in the dataset (the sample size was 2%). Both algorithms scale linearly, however SP performs significantly better than BEA.

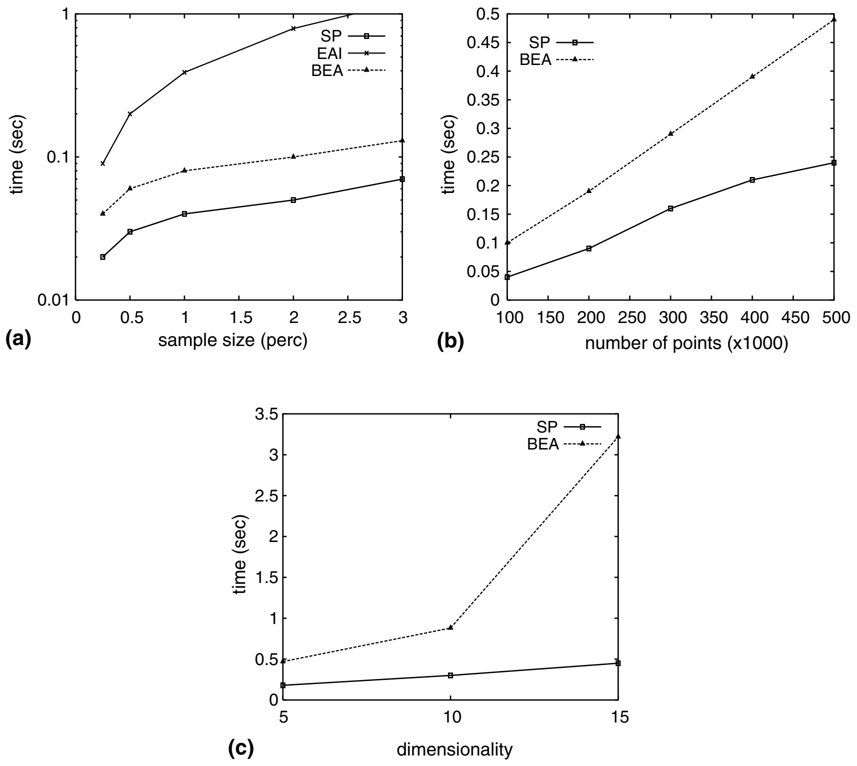

Fig. 9. Comparison of R-tree based DBS algorithms.

Next, we examine the impact of dimensionality. We used synthetical data sets that contained 100K points. We varied the dimensionality and the other parameters had the same values as those in previous experiments (notice that the page size is constant, thus the node capacity reduces with increasing dimensionality). Fig. 9c depicts the results. For lower dimensions, the performance difference between SP and BEA is similar to the results presented in the previous experiments. Nevertheless, the impact of larger dimensionality is more noticeable for BEA, that is clearly outperformed by SP.

### *7.3.2. Comparison with BS*

We now present the comparison of SP (the best R-tree based DBS algorithm) with BS. We focus on two separate cases: (1) when density information is not pre-computed, and (2) when density information is pre-computed. Regarding BS, case 1 corresponds to the computation of density estimator and its application during DBS, whereas case 2 corresponds only to the application of the pre-computed density estimator during DBS. Similar for SP, case 1 corresponds to the building of the R-tree and the application of the computed density estimation during DBS, whereas case 2 corresponds to the application of DBS over an existing R-tree.

Based on the above, we first compare SP with BS for case 1. We used the bulk-loading algorithm of [14] and 2-dimensional synthetic data sets that were analogous to those described in the previous experiments. For the measurement of the total execution time we included the index creation time for SP (the sorting phase of the bulk-loading algorithm is included) and the computation of the density estimator for BS. The cost for the sorting phase of [14] and the cost for the data set scans for BS (more than one in this case) depend on the amount of buffer memory. For this reason we examined two buffer sizes, i.e., 5% and 20% of the data set size. The results with respect to the data set size are depicted in Fig. 10a for buffer size equal to 5% and in Fig. 10b for buffer size equal to 20%. As shown, SP outperforms BS in both cases (it has to be noticed that for BS, the CPU cost is also a significant factor in this case, due to the computation of density information for each point). As expected, comparing the results between the two figures, the increase in buffer size results to a relative reduction in the execution times for both algorithms. Moreover, the execution time of SP depends solely on the time required for the index creation.

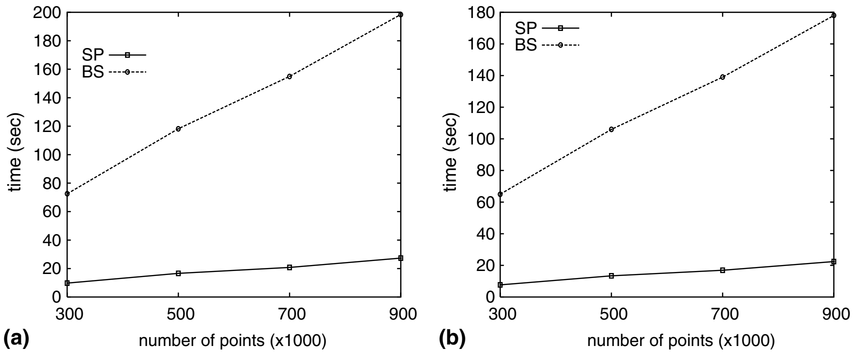

Fig. 10. Comparison of SP and BS for the case where information of density is not pre-computed: (a) buffer size 5% and, (b) buffer size 20%.

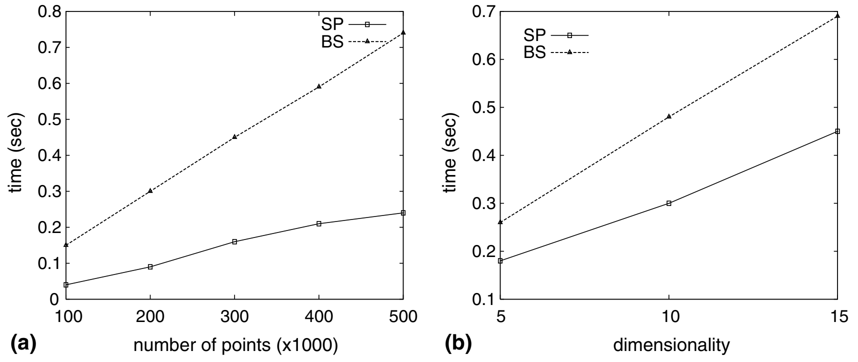

Fig. 11. Comparison of SP and BS for the case where density information is pre-computed.

Evidently, the use of a better bulk-loading method can further improve the results for SP. Nevertheless, the results in Fig. 10 indicate that SP is a viable solution, even in this case.

We now move on to case 2. For BS this means that the density estimation is stored (i.e., pre-computed) with each point, so as to avoid its recalculation during DBS. Equivalently (for fair comparison), SP uses an existing R-tree. Fig. 11a illustrates the results with respect to the dataset size. As shown, BS is clearly outperformed by SP in all cases. The reason is that BS has to examine all points (one scan over the entire dataset) in order to decide their inclusion in the sample. In addition it has to read the pre-stored estimations (one for each point in the dataset) to compute parameter $k$ (which cannot be pre-computed because it depends on $a$ that is a user-defined parameter—see Eq. (2)). In contrast, SP performs a selective pass, which avoids the reading of leaves that will not contribute to the sample. Moreover, the density information for SP (see Eq. (8)) is maintained in $I_L$ only for leaves instead of each single point (i.e., $|L| \ll N$); thus it requires much less CPU cost for this purpose.

Finally (for case 2), we tested the impact of dimensionality on SP and BS. Fig. 11b 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. The reason is that, as dimensionality increases, the scanning of each point by BS incurs higher cost.

## 7.4. Discussion

In this section, we summarize the presented experimental results and give descriptions for their better understanding. The results for the real data set (North East) illustrate that SP and BS lead to correct clustering of this data set, in contrast to GBS and US. This is also indicated by most of the cases for the synthetic data sets. Therefore, it can be concluded that SP and BS present an analogous effectiveness in terms of producing correct clustering results, and significantly outperform GBS and US. The slight improvement that SP presents over BS in some cases, is explained by the characteristics of the examined data sets. In particular: (a) The data sets have very high skew in cluster sizes (e.g., the *One Big* dataset [22]), a case that was not examined in [15], which for the examined values of parameters yield to correct clustering at smaller sample size. (b) Although the R-tree has not, per se, the objective of preserving densities, for data sets containing clusters (as the ones examined in this paper) this is attained at a good degree, because the cluster points are concentrated and can be well organized within the R-tree nodes. Therefore, due to the above cases, it can be stated that the objective of SP is to lead to results with quality analogous to those of kernel-density estimation methods (like the one in [15]) and to exploit, when possible, the characteristics of the data that suit well with the ones of the R-tree structure.

It has to be noticed that, for purposes of selectivity estimation in spatial databases, Acharya et al. [1] describe several histogram-based methods. They indicate that equi-area histograms lead to improved selectivity estimation compared to R-tree-based ones. However, the objectives of DBS (for clustering applications) are different than those of selectivity estimation, because DBS is only interested in detecting areas with high local density so as to focus the sampling on them and not to accurately estimate the number of points in the result of queries. More significantly, although GBS uses an analogous method to equi-area histograms (by using equi-area bins), the use of Eq. (1) may bias the sampling towards the areas that contain noise (the same conclusion is also indicated in [15]). Therefore, for data sets with very low noise (see Table 4a), GBS is effective, since Eq. (1) favors the small clusters only. In contrast, medium and high noise affect GBS significantly. Nevertheless, we address as an interesting topic of future work the development of DBS algorithms that will exploit histogram-based methods (e.g., MinSkew [1]).

Regarding the dimensionality of the data, previous research has indicated that the performance of R-tree-like indexes is affected by high dimensionality [5]. In the cases examined herein (Table 4b), SP performed well for low and medium dimensions (less than 8), whereas it produced correct clustering results for some more dimensions (10–12). The reason for the latter case is that the increase in dimensionality is compensated by the fact that data are well clustered and can be organized in a good way within the R-tree nodes (this cannot be well achieved for not clustered data of such dimensionality). However, for more high dimensions (e.g., 15), the performance of SP is affected. Therefore, for high dimensionality, the proposed methodology should be examined with other tree structures (e.g., the X-tree [5], which has some similarities with R-tree structures). We address this issue as interesting future work.

Focusing on the results on efficiency, we must notice that R-tree-based methods (SP, BEA, EAI) use random access to leaves, whereas the others (BS, GBS, US) use scanning of sequential files. As described (see also Fig. 7b), SP avoids the access to a large number of leaves (for 1% sample size it accesses less than 50% of leaves). Therefore, one may consider that there is a kind of trade-off between the retrieval of a number of leaves and the sequential scanning of a file, depending on how scattered are the disk pages of the index or of the sequential file. Nevertheless, the placement and management of disk pages of the the R-tree leaves or of the sequential file, depend on the file-system that is used and on other related factors (e.g., amount of disk fragmentation, if the index is bulk-loaded, internal fragmentation within pages). Therefore, a generalized conclusion cannot be easily obtained. For this reason, we measured the total execution time when comparing the different methods. Another reason for the latter choice is that, besides I/O, CPU time is also a significant factor. Based on our observation through the experiments, BS in particular requires high CPU time, due to the examination of each point. In contrast, as described, SP examines the density estimation on a node basis, leading to smaller CPU time.

In summary, the main objective of the experimental results was to show that the proposed approach can attain samples of good quality with low execution times. The focus in the development of SP was on how to perform DBS from spatial indexes. Therefore, the examination of existing

DBS methods (like BS or GBS) was done for purposes of comparison and the target was not to replace such methods, since (as shown) they can be used with efficacy in several cases.[13] Moreover, it should not be interpreted from the experimental results that Uniform sampling is worthless; only that its use may impact the effectiveness in case of clusters with skew sizes, which is the target of DBS.

## 8. 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 good quality and with low execution time for the sampling procedure.

We described SP, a novel DBS algorithm. SP, differently from adaptations that were derived based on the paradigm of uniform sampling from indexes, considers directly the partitioning of data into the nodes of the spatial index. Therefore, it does not present the overhead of having to compensate the differences in the fanouts of the upper-level nodes. An optimized version of SP can further improve the performance, by avoiding the reading of nodes that will not contribute to the final sample.

We formally showed that SP produces samples that guarantee the criteria required by DBS. We also derived formulae for the cost of SP (and of the adapted algorithms) and performed an analytical performance comparison, which illustrates the superiority of the proposed algorithm. By using synthetic and real data, we also performed an experimental comparison with prior algorithms (BS, GBS, and US). Our experiments, which considered a variety of factors, illustrated that SP can produce samples of good quality and has low execution times.

Conclusively, with the proposed approach, spatial indexes comprise an effective and efficient *sampling frame* [3] for DBS. Finally, we argue that the proposed approach is easily extendible to the task of outlier detection using samples [15], since in this case sampling with probability *inversely* proportional to density is required. We address this issue as future work.

## References

[1] S. Acharya, V. Poosala, S. Ramaswamy, Selectivity estimation in spatial databases, in: Proceedings of the ACM Conference on Management of Data (SIGMOD'99), 1999, pp. 13–24.

[2] G. Antoshenkov, Random sampling from pseudo-ranked $B^+$ trees, in: Proceedings of the Conference on Very Large Databases (VLDB'92), 1992, pp. 375–382.

[3] D. Barbara, C. Faloutsos, J. Hellerstein, Y. Ioannidis, H.V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. Ross, K.C. Sevcik, The New Jersey data reduction report, IEEE Data Eng. Bull. 20 (4) (1997) 3–45.

[4] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger, The R*-tree: an efficient and robust access method for points and rectangles, in: Proceedings of the ACM Conference on Management of Data (SIGMOD'90), 1990, pp. 322–331.

[13] Analogously, [20] was not proposed to substitute reservoir sampling methods [30], but to provide a method for sampling when data are indexed instead of being stored in a sequential file.

[5] S. Berchtold, D. Keim, H.-P. Kriegel, The X-tree: an index structure for high-dimensional data, in: Proceedings of the Conference on Very Large Databases (VLDB'96), 1996, pp. 28–39.
[6] S. Brakatsoulas, D. Pfoser, Y. Theodoridis, Revisiting R-tree construction principles, in: Proceedings of the Conference on Advances in Databases and Information Systems (ADBIS'02), 2002.
[7] M. Breunig, H.-P. Kriegel, P. Kroger, J. Sander, Data bubbles: quality preserving performance boosting for hierarchical clustering, in: Proceedings of the ACM Conference on Management of Data (SIGMOD'01), 2001, pp. 79–90.
[8] M. Ester, H.-P. Kriegel, X. Xu, Knowledge discovery in large spatial databases: focusing techniques for efficient class identification, in: Proceedings of the Symposium on Large Spatial Databases (SSD'95), 1995, pp. 67–82.
[9] P. Gibbons, Y. Matias, V. Poosala, Fast incremental maintenance of approximate histograms, in: Proceedings of the Conference on Very Large Databases (VLDB'97), 1997, pp. 466–475.
[10] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: Proceedings of the ACM Conference on Management of Data (SIGMOD'84), 1984, pp. 47–57.
[11] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for large databases, in: Proceedings of ACM Conference on Management of Data (SIGMOD'98), 1998, pp. 73–84.
[12] A. Hinneburg, D. Keim, Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering, in: Proceedings of the Conference on Very Large Databases (VLDB'99), 1999, pp. 506–517.
[13] G. John, P. Langley, Static versus dynamic sampling for data mining, in: Proceedings of the Conference on Knowledge Discovery and Data Mining (KDD'96), 1996, pp. 367–370.
[14] I. Kamel, C. Faloutsos, On packing R-trees, in: Proceedings of International Conference on Information and Knowledge Management (CIKM'93), 1993, 490–499.
[15] G. Kollios, D. Gunopoulos, N. Koudas, S. Berchtold, Efficient biased sampling for approximate clustering and outlier detection in large datasets, IEEE Trans. Knowledge Data Eng. (TKDE) (to appear).
[16] J. Kivinen, H. Mannila, The power of sampling in knowledge discovery, in: Proceedings of ACM Symposium on Principles of Database Systems (PODS'94), 1994, pp. 77–85.
[17] 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 2 (3) (1998) 233–262.
[18] C. Lang A. Singh modeling high-dimensional index structures using sampling, in: Proceedings of ACM Conference on Management of Data (SIGMOD'01), 2001, pp. 389–400.
[19] A. Nanopoulos, Y. Theodoridis, Y. Manolopoulos, $C^2P$: clustering based on closest pairs, in: Proceedings of the Conference on Very Large Databases (VLDB'01), 2001, 331–340.
[20] F. Olken, D. Rotem, Random sampling from $B^+$-trees, in: Proceedings of the Conference on Very Large Databases (VLDB'89), 1989, pp. 269–277.
[21] F. Olken, D. Rotem, Sampling from spatial databases, in: Proceedings of IEEE Conference on Data Engineering (ICDE'93), 1993, pp. 199–208.
[22] C. Palmer, C. Faloutsos, Density biased sampling: an improved method for data mining and clustering, in: Proceedings of ACM Conference on Management of Data (SIGMOD'00), 2000, pp. 82–92.
[23] B.-U. Pagel, H.-W. Six, H. Toben, P. Widmayer, Towards an analysis of range query performance in spatial data structures, in: Proceedings of ACM Symposium on Principles of Database Systems (PODS'93), 1993, pp. 214–221.
[24] S. Ravada, J. Sharma, Oracle8i spatial: experiences with extensible databases, in: Proceedings of International on Large Spatial Databases (SSD'99), 1999, pp. 355–359.
[25] T. Reinartz, Similarity-driven sampling for data mining, in: Proceedings of Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98), 1998, pp. 423–431.
[26] N. Roussopoulos, S. Kelley, F. Vincent, Nearest neighbor queries, in: Proceedings of ACM Conference on Management of Data (SIGMOD'95), 1995, 71–79.
[27] Y. Theodoridis, T. Sellis, A model for the prediction of R-tree performance, in: Proceedings of ACM Symposium on Principles of Database Systems (PODS'96), 1996, pp. 161–171.
[28] H. Toivonen, Sampling large databases for association rules, in: Proceedings of the Conference on Very Large Databases (VLDB'96), 1996, pp. 134–145.
[29] M. Vassilakopoulos, Y. Manolopoulos, On sampling regional data, Data Knowledge Eng. (DKE) 22 (3) (1997) 309–318.

[30] J.S. Vitter, Random sampling with a reservoir, ACM Trans. Math. Software 11 (1) (1985) 37–57.
[31] M. Zaki, S. Parthasarathy, W. Li, M. Ogihara, Evaluation of sampling for data mining of association rules, in: Proceedings of Workshop on Research Issues in Data Engineering (RIDE'97), 1997.
[32] S. Zhou, A. Zhou, J. Cao, J. Wen, Y. Fan ,Y. Hu, Combining sampling technique with DBSCAN algorithm for clustering large spatial databases, in: Proceedings of the Conference on Knowledge Discovery and Data Mining (PAKDD'00), 2000, pp. 169–172.

**Alexandros Nanopoulos** was born in 1974. He graduated from the Department of Informatics, Aristotle University of Thessaloniki, Greece, on November 1996, and obtained a Ph.D. from the same institute, on February 2003. The subject of his dissertation was: "Techniques for Non Relational Data Mining". He is co-author of more than 20 articles in international journals and conferences, also co-author of the monograph "Advanced Signature Techniques for Multimedia and Web Applications". His research interests include spatial and web mining, integration of data mining with DBMSs, and spatial database indexing.

**Yannis Theodoridis** was born in 1967, received his Diploma (1990) and Ph.D. (1996) in Electrical and Computer Engineering, both from the National Technical University of Athens, Greece. Since January 2002, he is Assistant Professor at the Department of Informatics, University of Piraeus, and has also a joint research position at the Computer Technology Institute (CTI). His research interests include spatial and spatiotemporal databases, location-based data management, data mining and geographical information systems. He is co-author of the book "Advanced Database Indexing" (1999, Kluwer Academic Publishers) and has published over 30 articles in scientific journals such as Algorithmica, ACM Multimedia, IEEE Transactions in Knowledge and Data Engineering, and in conferences such as ACM SIGMOD, PODS, ICDE, VLDB. His work has over 250 citations in scientific journals and conference proceedings. He has served in the program committee for several conferences (SIGMOD, ICDE, SSTD, etc.) and in the editorial board of International Journal of Data Warehousing and Mining (IJDWM). He was general chair for the 8th International Symposium on Spatial and Temporal Databases (SSTD'03). He is member of ACM and IEEE.

**Yannis Manolopoulos** was born in Thessaloniki, Greece in 1957. He received a B.Eng. (1981) in Electrical Engineering and a Ph.D. (1986) in Computer Engineering, both from the Aristotle Univ. of Thessaloniki. Currently, he is Professor at the Department of Informatics of the latter university. He has been with the Department of Computer Science of the University of Toronto, the Department of Computer Science of the University of Maryland at College Park and the University of Cyprus. He has published over 130 papers in refereed scientific journals and conference proceedings. He is co-author of a book on "Advanced Database Indexing" and "Advanced Signature Indexing for Multimedia and Web Applications" by Kluwer. He served/serves as PC Co-chair of the 8th National Computer Conference (2001), the 6th ADBIS Conference (2002) the 5th WDAS Workshop (2003), the 8th SSTD Symposium (2003) and the 1st Balkan Conference in Informatics (2003), the 16th SSDBM Conference (2004). Also, currently he is Vice-chairman of the Greek Computer Society. His research interests include databases, data mining, information retrieval, and performance evaluation of storage subsystems. Further information can be found at http://delab.csd.auth.gr.