devarapalli.demudubabu, Gachi swami/ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 1, Issue 3, pp.1189-1199

Fast QueryProcessing With Clustering Algorithm

devarapalli.demudubabu(m.tech)
jyothismathi engineering college,karimnagar, Andhrapradesh, india

Gachi swami(associate professor)
Jyothismathi engineering college, karinnagar, Andhrapradesh,india

Abstract

Clustering, in data mining, is useful for discovering groups and identifying interesting distributions in the underlying data and to fast query processing. Traditional clustering algorithms either favor clusters with spherical shapes and similar sizes, or are very fragile in the presence of outliers. We propose a new clustering algorithm called CURE that is more robust to outliers, and identifies clusters having non-spherical shapes and wide variances in size. CURE achieves this by representing each cluster by a certain fixed number of points that are generated by selecting well scattered points from the cluster and then shrinking them toward the center of the cluster by a specified fraction. Having more than one representative point per cluster allows CURE to adjust well to the geometry of non-spherical shapes and the shrinking helps to dampen the effects of outliers. To handle large databases, CURE employs a combination of random sampling and partitioning. A random sample drawn from the data set is first partitioned and each partition is partially clustered. The partial clusters are then clustered in a second pass to yield the desired clusters.Our experimental results confirm that the quality of clusters produced by CURE is much better than those found by existing algorithms. Furthermore, they demonstrate that random sampling and partitioning enable CURE to not only outperform existing algorithms but also to scale well for large databases without sacrificing clustering quality.

Keywords:randomsampling,partitioning,clustering algorithm.

1 Introduction

The wealth of information embedded in huge databases belonging to corporations (e.g., etail, financial, telecom) has spurred a tremendous interest in the areas of lcnowledge discovery and data mining. Clustering, in data mining, is a useful technique for discovering interesting data distributions and patterns in the underlying data. The problem of clustering can be defined as follows: given n data points in a d-dimensional metric space, partition the data points into small points to make fast searching of data.

Fast QueryProcessing With Clustering Algorithm

clusters such that the data points within a cluster are more similar to each other than data points in different clusters.

2. Traditional Clustering Algorithms - Drawbacks

Existing clustering algorithms can be broadly classified into partitional and hierarchical [JD88]. Partitional clustering algorithms attempt to determine k partitions that optimize a certain criterion function. The square-error criterion, defined below, is the most commonly used (m; is the mean of cluster Ci).

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} \|p - m_i\|^2.$$

The square-error is a good measure of the within-cluster variation across all the partitions. The objective is to find L partitions that minimize the square-error. Thus, squareerror clustering tries to make the k clusters as compact and separated as possible, and works well when clusters are compact clouds that are rather well separated from one another. However, when there are large differences in the sizes or geometries of different clusters, as illustrated in Figure 1, the square-error method could split large clusters to minimize the square-error. In the figure, the square-error is larger for the three separate clusters in (a) than for the three clusters in (b) where the big cluster is split into three portions, one of which is merged with the two smaller clusters. The reduction in square-error for (b) is due to the fact that the slight
reduction in square error due to splitting the large cluster is weighted by many data points in the large cluster. A hierarchical clustering is a sequence of partitions in which each partition is nested into the next partition in the sequence. An agglomerative algorithm for hierarchical clustering starts with the disjoint set of clusters, which places each input data point in an individual cluster. Pairs of items or clusters are then successively merged until the number of clusters reduces to Ic. At each step, the pair of clusters merged are the ones between which the distance is the minimum. The widely used measures for distance between clusters are as follows (mi is the mean for cluster Ci and n;
is the number of points in Ci).

$$d_{mean}(C_i, C_j) = \|m_i - m_j\|$$

$$d_{ave}(C_i, C_j) = 1/(n_i n_j) \sum_{p \in C_i} \sum_{p' \in C_j} \|p - p'\|$$

$$d_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \|p - p'\|$$

Fast QueryProcessing With Clustering Algorithm

Figure 1: Splitting of a large cluster by partitional algorithms

For example, with d,,,, as the distance measure, at each step, the pair of clusters whose centroids or means are the closest are merged. On the other hand, with d,i,, the pair of clusters merged are the ones containing the closest pair of points. All of the above distance measures have a minimum variance flavor and they usually yield the same results if the

clusters are compact and well-separated. However, if the clusters are close to one another (even by outliers), or if their shapes and sizes are not hyperspherical and uniform, the results

of clustering can vary quite dramatically. For example, with the data set shown in Figure l(a), using d,,,, d,,, or d,,,, as the distance measure results in clusters that are similar to those obtained by the square-error method shown in Figure l(b). Similarly, consider the example data points in Figure 2. The desired elongated clusters are shown in Figure 2(a). However, d,,,, as the distance measure, causes the elongated clusters to be split and portions belonging to neighboring elongated clusters to be merged. The resulting clusters are as shown in Figure 2(b). On the other hand, with dmin as the distance measure, the resulting clusters are as shown in Figure 2(c). The two elongated clusters that are connected by narrow string of points are merged into a single cluster. This “chaining effect” is a drawback of d,in - basically, a few points located so as to form a bridge between the two clusters causes points across the clusters to be grouped into a single elongated cluster. From the above discussion, it follows that neither the centroid-based approach (that uses d,,,,) nor the all-points approach (based on d,i,) work well for non-spherical or arbitrary shaped clusters. A shortcoming of the centroidbased approach is that it considers only one point as representative of a cluster - the

cluster centroid. For a large or arbitrary shaped cluster, the centroids of its subclusters can be reasonably far apart, thus causing the cluster to be split. The all-points approach, on the other hand, considers all the points within a cluster as representative of the cluster. This other extreme, has its own drawbacks, since it makes the clustering algorithm extremely sensitive to outliers and to slight changes in the position of data points. When the number N of input data points is large, hierarchical clustering algorithms break down due to their non-linear time complexity (typically, O(N’)) and huge I/O costs. In order to remedy this problem, in [ZRL96], the authors propose a new clustering method named BIRCH, which represents the state of the art for clustering large data sets.

Fast QueryProcessing With Clustering Algorithm

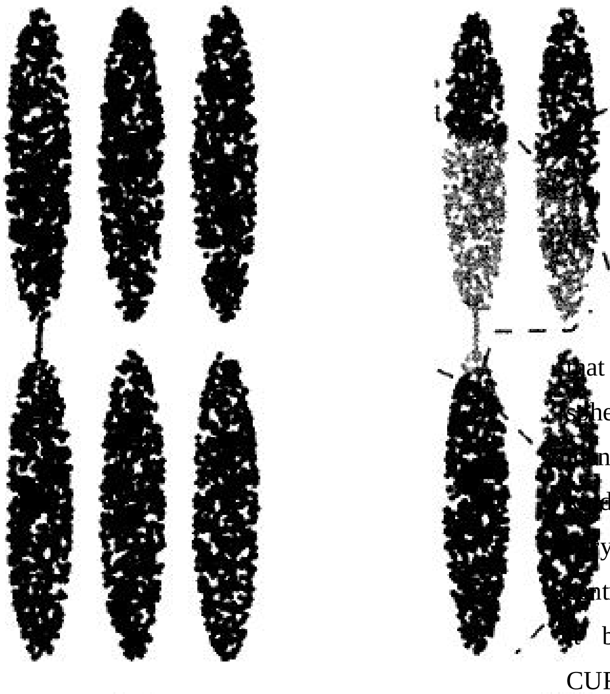
BIRCH first performs a preclzlstering phase in which dense regions of points are represented by compact summaries, and then a centroid-based hierarchical algorithm is used to cluster the set of summaries (which is much smaller than the original dataset). The preclustering algorithm employed by BIRCH to reduce input size is incremental and approximate. During preclustering, the entire database is scanned, and cluster summaries are stored in memory in a data structure called the CF-tree. For each successive data point, the CF-tree is traversed to find the closest cluster to it in the tree, and if
the point is within a threshold distance of the closest cluster, it is absorbed into it. Otherwise, it starts its own cluster in the CF-tree. Once the clusters are generated, a final labeling phase is carried out in which using the centroids of clusters as seeds, each data point is assigned to the cluster with the closest seed. Using only the centroid of a cluster when redistributing the data in the final phase has problems when clusters do not have uniform sizes and shapes as in Figure 3(a). In this case, as illustrated in Figure 3(b), in the final labeling phase, a number of points in the bigger cluster are labeled as belonging to the smaller cluster since they are closer to the centroid of the smaller cluster.

3.Our Contributions

In this paper, we propose a new clustering method named CURE (Clustering Using Representatives) whose salient features are described below. Hierarchical Clustering Algorithm: CURE employs a novel hierarchical clustering algorithm that adopts a middle ground between the centroid-based and the all-point extremes. In CURE, a constant number c of well scattered points in a cluster are first chosen. The scattered points capture the shape and extent of the cluster. The chosen scattered points are next shrunk towards the centroid of the cluster by a fraction cr. These scattered points after shrinking are used as representatives
of the cluster. The clusters with the closest pair of representative points are the clusters that are merged at each step of CURE's hierarchical clustering algorithm. The scattered points approach employed by CURE alleviates the shortcomings of both the all-points as well as the centroid-based approaches. It enables CURE to correctly identify the clusters in Figure 2(a) - the resulting clusters due to the centroid-based and all-points approaches is as shown in Figures 2(b) and 2(c), respectively. CURE is less sensitive to outliers since shrinking the scattered points toward the mean dampens the adverse effects due to outliers ~ outliers are typically further away from the mean and are thus shifted a larger distance due to the shrinking. Multiple scattered points also enable CURE to discover non-spherical clusters like the elongated clusters shown in Figure 2(a). For the centroid-based algorithm, the space that constitutes the vicinity of the single centroid for a cluster is spherical.

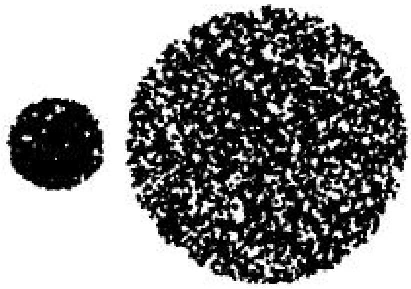
Fast QueryProcessing With Clustering Algorithm

Thus, it favors spherical clusters and as shown in Figure 2(b), splits the elongated clusters. On the other hand, with multiple scattered points as representatives of a cluster, the space

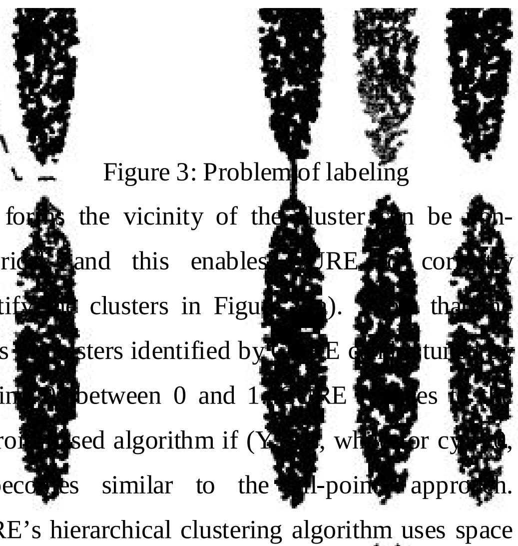


(a)

Figure 2: Clusters generated by hierarchical algorithms



(a)



(b) (c)

Figure 3: Problem of labeling

that forms the vicinity of the cluster can be non-spherical and this enables CURE to correctly identify the clusters in Figure 2(a). Note that the kinds of clusters identified by CURE can be tuned by varying α between 0 and 1. CURE reduces to the centroid-based algorithm if (Y = 1), while for cy = 0, it becomes similar to the all-points approach. CURE's hierarchical clustering algorithm uses space that is linear in the input size n and has a worst-case time complexity of O(n2 log n). For lower dimensions (e.g., two), the complexity can be shown to further reduce to O(n'). Thus, the time complexity of CURE is no worse than that of the centroidbased hierarchical algorithm.

Random Sampling and Partitioning:

CURE's approach to the clustering problem for large data sets differs from BIRCH in two ways. First, instead of preclustering with all the data points,

Fast QueryProcessing With Clustering Algorithm

CURE begins by drawing a random sample from the database. We show, both analytically and experimentally, that random samples of moderate sizes preserve information about the geometry of clusters fairly accurately, thus enabling CURE to correctly cluster the input. In particular, assuming that each cluster has a certain minimum size, we use chernoff bounds to calculate the minimum sample size for which the sample contains, with high probability, at least a fraction f of every cluster. Second, in order to further speed up clustering, CURE first partitions the random sample and partially clusters the data points in each partition. After eliminating outliers, the preclustered data in each partition is then clustered in a final pass to generate the final clusters.

Labeling Data on Disk:

Once clustering of the random sample is completed, instead of a single centroid, multiple representative points from each cluster are used to label the remainder of the data set. The problems with BIRCH's labeling phase are eliminated by assigning each data point to the cluster containing the closest representative point. Overview: The steps involved in clustering using CURE.

Clustering procedure: Initially, for each cluster u, the set of representative points w..rep contains only the point in the cluster. Thus, in Step 1, all input data points are inserted into the k-d tree. The procedure buildheap (in Step 2) treats each input point as a separate cluster, computes uclosest for each cluster u and then inserts each cluster into the heap (note that the clusters are arranged in the increasing order of distances between u and wclosest). Once the heap Q and tree T are initialized, in each iteration of the while-loop, until only k clusters remain, the closest pair of clusters is merged. The cluster u at the top of the heap Q is the cluster for which u and uxlosest are the closest pair of clusters. Thus, for each step of the whileloop, extractmin (in Step 4) extracts the top element u in

Q and also deletes u from Q.

```
procedure cluster(S, k)
begin
1.  T := build_kd_tree(S)
2.  Q := build_heap(S)
3.  while size(Q) > k do {
4.     u := extract_min(Q)
5.     v := u.closest
6.     delete(Q, v)
7.     w := merge(u, v)
8.     delete_rep(T, u); delete_rep(T, v); insert_rep(T, w)
9.     w.closest := x /* x is an arbitrary cluster in Q */
10.    for each x ∈ Q do {
11.       if dist(w, x) < dist(w, w.closest)
12.          w.closest := x
13.       if x.closest is either u or v {
14.          if dist(x, x.closest) < dist(x, w)
15.             x.closest := closest_cluster(T, x, dist(x, w))
16.          else
17.             x.closest := w
18.          relocate(Q, x)
19.       }
20.       else if dist(x, x.closest) > dist(x, w) {
21.          x.closest := w
22.          relocate(Q, x)
23.       }
24.    }
25.    insert(Q, w)
26. }
end
```

Figure 4: Clustering algorithm

Fast QueryProcessing With Clustering Algorithm

4. Clustering Algorithm

In this subsection, we describe the details of our clustering algorithm(seeFigure 5). The input parameters to our algorithm are the input data set S containing n points in ddimensional space and the desired number of clusters k. As we mentioned earlier, starting with the individual points as individual clusters, at each step the closest pair of clusters is merged to form a new cluster. The process is repeated until there are only k remaining clusters.

5.Algorithms

BIRCH: We used the implementation of BIRCH provided to us by the authors of [ZRL96]. The mplementation performs preclustering and then uses a centroid-based hierarchical clustering algorithm with time and space complexity that is quadratic in the number of points after preclustering. We set parameter values to the default values suggested in [ZRL96]. For example, we set the page size to 1024 bytes and the input size to the hierarchical clustering algorithm after the preclustering phase to 1000. The memory used for preclustering was set to be about 5% of dataset size.

CURE: Our version of CURE is based on the clustering algorithm described in Section 4, that uses representative points with shrinking towards the mean. As described at the end of Section 3, when two clusters are merged in each step of the algorithm, representative points for the new merged cluster are selected from the ones for the two original clusters rather than all points in the merged cluster. This improvement speeds up execution times for CURE without adversely impacting the quality of clustering.

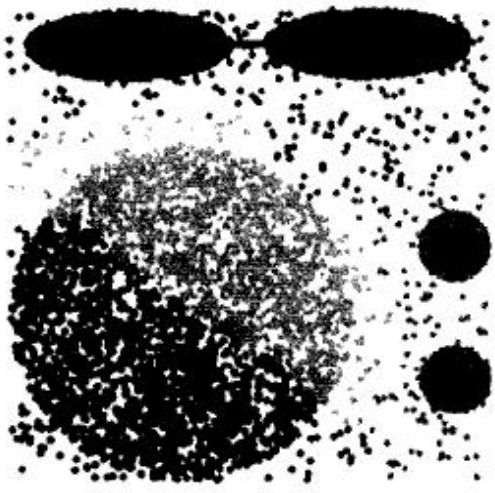
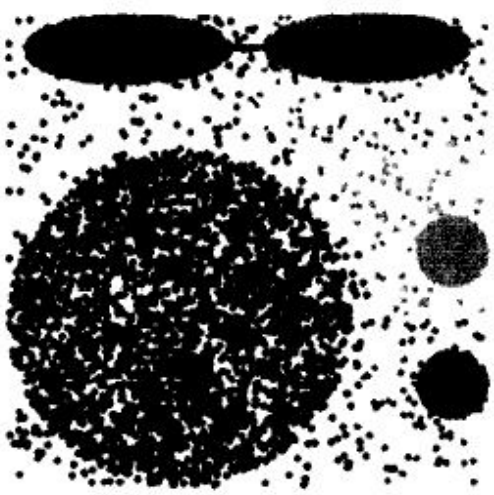


figure 5 datasets (b) CURE

Fast QueryProcessing With Clustering Algorithm



(a) BRICH

Figure: 5 datasets

5.1.Time and Space Complexity

The worst-case time complexity of our clustering algorithm can be shown to be O(n2 logn). In [GRS97], we show that when the dimensionality of data points is small, the time complexity further reduces to O(n2). Since both the heap and the /c-d tree require linear space, it follows that the spacecomplexity of our algorithm is O(n).

6. Enhancements for Large Data Sets

Most hierarchical clustering algorithms, including the one presented in the previous subsection, cannot be directly applied to large data sets due to their quadratic time complexity with respect to the input size. In this section, we present enhancements and optimizations that enable CURE to handle large data sets. We also address the issue of outliers and propose schemes to eliminate them.

6.1 Random Sampling

In order to handle large data sets, we need an effective mechanism for reducing the size of the input to CURE's clustering algorithm. One approach to achieving this is via random sampling - the key idea is to apply CURE's clustering algorithm to a random sample drawn from the data set rather than the entire data set. Typically, the random sample will fit in main-memory and will be much smaller than the original data set. Consequently, significant improvements in execution times for CURE can be realized. Also, random sampling can improve the quality of clustering since it has the desirable effect of filtering outiiers. Efficient algorithms for drawing a sample randomly from data in a file in one pass and using constant space are proposed in [Vit85]. As a result, we do not discuss sampling in any further detail, and assume that we employ one of the well-known algorithms for generating the random sample. Also, our experience has been that generally, the overhead of generating a random sample is very small compared to the time for performing clustering on the sample (the random sampling algorithm typically takes less than two seconds to sample a few thousand points from a file containing hundred thousand or more points).

6.2 Partitioning for Speedup

Fast QueryProcessing With Clustering Algorithm

As the separation between clusters decreases and as clusters become less densely packed, samples of larger sizes are required to distinguish them. However, as the input size n grows, the computation that needs to be performed by CURE's clustering algorithm could end up being fairly substantial due to the O(n2 log n) time complexity. In this subsection, we propose a simple partitioning scheme for speeding up CURE when input sizes become large.

The basic idea is to partition the sample space into p partitions, each of size %. We then partially cluster each partition until the final number of clusters in each partition reduces to 5 for some constant q > 1. Alternatively, we could stop merging clusters in a partition if the distance between the closest pair of clusters to be merged next increases above a certain threshold. Once we have generated $ clusters for each partition, we then run a second clustering pass on the s partial clusters for all the partitions (that resulted from the first pass).

6.3 Labeling Data on Disk

Since the input to CURE's clustering algorithm is a set of randomly sampled points from the original data set, the final k clusters involve only a subset of the entire set of points. In CURE, the algorithm for assigning the appropriate cluster labels to the remaining data points employs a fraction of randomly selected representative points for each of the final k clusters. Each data point is assigned to the cluster containing the representative point closest to it. Note that approximating every cluster with multiple points instead a single centroid as is done in [ZRL96], enables CURE to, in the final phase, correctly distribute the data points when clusters are non-spherical or non-uniform. The final labeling phase of [ZRLSG], since it employs only the centroids of the clusters for partitioning the remaining points,

6.4 Handling Outliers

Any data set almost always contains o&hers. These do not belong to any of the clusters and are typically defined to be points of non agglomerative behavior. That is, the neighborhoods

of outliers are generally sparse compared to points in clusters, and the distance of an outlier to the nearest cluster is comparatively higher than the distances among points in bon&de clusters themselves. Every clustering method needs mechanisms to eliminate outliers. In CURE, outliers are dealt with at multiple steps. First, random sampling filters out a majority of the outliers. Furthermore, the few outlicrs that actually make it into the random sample are distributed all over the sample space. Thus, random sampling further isolates outliers. In agglomerative hierarchical clustering, initially each point is a separate cluster. Clustering then proceeds by merging closest points first. What this suggests is that outliers, due to their larger distances from other points, tend to merge with

other points less and typically grow at a much slower rate than actual clusters. Thus, the number of points in a collection of outliers is typically much less than the number in a cluster.

7.conclusion

In this paper, we addressed problems with traditional clustering algorithms and solution to enhance the query processing.which either favor clusters with spherical shapes and similar sizes, or are very fragile in the presence of outliers. We proposed a clustering method called CURE. CURE utilizes multiple representative points for each cluster that are generated by selecting well scattered points from the cluster and then shrinking them toward the center of the cluster by a specified fraction. This enables CURE to adjust well to the geometry of clusters having non-spherical shapes and wide variances in size. To handle large databases, CURE employs a combination of random sampling and partitioning that allows it to handle large data sets efficiently. Random sampling, coupled with outlier handling techniques, also makes it possible for CURE to filter outliers contained in the data set effectively. Furthermore, the labeling algorithm in CURE uses multiple random representative points or each cluster to assign data points on disk. This enables t to correctly label points even when the shapes of clusters are non-spherical and the sizes of clusters vary. For a random sample size of 3, the time complexity of CURE is 0(s2) for low-dimensional data and the space complexity is linear in s. To study the effectiveness of CURE for clustering large data sets, we conducted extensive experiments. Our results confirm that the quality of clusters produced by CURE is much better than those found by existing algorithms. Furthermore, they demonstrate that CURE not only outperforms existing algorithms but also scales well for large databases without sacrificing clustering quality.

REFERENCES

[1]N. Beckmann, H.-P. Kriegei, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In Proc. Of ACM SIGMOD, pages 322-331, Atlantic City, NJ,May 1990

[2]Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. Introduction to Algorithms. The MIT Press, Massachusetts, 1990.

[3]Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial database with noise. In I&‘1 Conference on Knowledge Discovery in Databases and Data Mining (KDD-96), Portland, Oregon, August 1996.

[4]Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. A database interface for clustering

Fast QueryProcessing With Clustering Algorithm

in large spatial databases. In Int'l Conference on Knowledge Discovery in Databases and Data Mining (KDD-95), Montreal, Canada, August 1995.

[5]Sudipto Guha, R. Rastogi, and K. Shim. CURE: A clustering algorithm for large databases. Technical report, Bell Laboratories, Murray Hill, 1997.

[6]Anil K. Jain and Richard C. Dubes. Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[7]R. Motwani and P. Raghavan. Randombed Algorithms. Cambridge University Press, 1995.

[8]Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In Proc. of the VLDB Conference, Santiago, Chile, September
1994.

[10]Clark F. Olson. Parallel algorithms for hierarchical clustering. Technical report, University of California at Berkeley, December 1993.

[11]H. Samet. The Design and Analysis of Spatial Data Structures. Addison-Wesley, 1989. Hanan Samet. The Design and Analysis of Spatial Data Structures. Addison-Wesley Publishing Company, Inc., New York, 1990.

[12]T. Sellis, N. Roussopoulos, and C. Faioutsos. The R+ tree: a dynamic index for multi-dimensional objects. In Proc. 13th Int'l Conference on VLDB, pages 507~- 518, England, 1987.

[13]Jeff Vitter. Random sampling with a reservoir. ACM Transactions on Mathematical Software, 11(1):37-57, 1985.

[14]Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In Proceedings of the ACM SIGMOD Conference on ManagementofData,pageS103-114, Montreal, Canada, June 1996.