Available online at www.sciencedirect.com
SCIENCE DIRECT®
The Journal of Systems and Software 73 (2004) 169–180
The Journal of Systems and Software
www.elsevier.com/locate/jss

A top-down approach for density-based clustering using multidimensional indexes

Jae-Joon Hwang [a,*], Kyu-Young Whang [a], Yang-Sae Moon [a], Byung-Suk Lee [b]

[a] Department of Computer Science and Advanced Information Technology Research Center, Korea Advanced Institute of Science and Technology, 373-1, Kusong-Dong, Yusong-Gu, Daejeon 305-701, South Korea
[b] Department of Computer Science, University of Vermont, Burlington, VT 05405, USA

Received 30 March 2003; received in revised form 24 August 2003; accepted 27 August 2003
Available online 25 December 2003

Abstract

Clustering on large databases has been studied actively as an increasing number of applications involve huge amount of data. In this paper, we propose an efficient top-down approach for density-based clustering, which is based on the density information stored in index nodes of a multidimensional index. We first provide a formal definition of the cluster based on the concept of *region contrast partition*. Based on this notion, we propose a novel top-down clustering algorithm, which improves the efficiency through branch-and-bound pruning. For this pruning, we present a technique for determining the bounds based on *sparse and dense internal regions* and formally prove the correctness of the bounds. Experimental results show that the proposed method reduces the elapsed time by up to 96 times compared with that of BIRCH, which is a well-known clustering method. The results also show that the performance improvement becomes more marked as the size of the database increases.

Keywords: Databases; Clustering; Density-based pruning; Multidimensional indexes

1. Introduction

Data mining has become a research area of increasing importance. In particular, clustering on a large database has become one of the most actively studied topics of data mining (Chen et al., 1996). Clustering, also known as unsupervised learning, distinguishes dense areas with high data concentration from sparse areas to find useful patterns of data distribution in the database (Ankerst et al., 1999; Ester et al., 1996; Karypis et al., 1999; Kaufman and Rousseeuw, 1990; Ng and Han, 1994; Schikuta, 1996). The purpose of clustering is to group the objects of a database into meaningful subclasses, called *clusters* (Ester et al., 1996). Clustering is widely used in various applications such as customer purchase pattern analysis, medical data analysis, geographical information analysis, and image analysis. An example is seismic fault detection in a geographic information system when provided with data on earthquakes in seismic regions (see <http://www.ceri.memphis.edu> for examples). Here, we can locate the faults by partitioning the data into two regions, one with frequent earthquakes and the other without, via clustering. Basically, the focus of clustering methods has been on the accuracy of clusters and the computation time. As databases become larger, however, most clustering methods are no longer practical because of excessive processing time. Therefore, recent clustering techniques are focusing on the scalability (Breunig et al., 2001; Ganti et al., 1999).

In this paper, we propose a novel top-down clustering approach that avoids such excessive computations by searching an index for densely populated regions in a database. In particular, this method takes advantage of a multidimensional index commonly used in large database applications (such as data warehouses and geographical information systems). In a multidimensional

* Corresponding author. Tel.: +82-42-869-5562; fax: +82-42-869-3510.
E-mail addresses: jjhwang@mozart.kaist.ac.kr (J.-J. Hwang), kywhang@mozart.kaist.ac.kr (K.-Y. Whang), ysmoon@mozart.kaist.ac.kr (Y.-S. Moon), bslee@cs.uvm.edu (B.-S. Lee).

0164-1212/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2003.08.237

index, objects that are closer to each other have a higher probability of being stored in the same or adjacent data pages. This is called the *clustering property* (Ester et al., 1995; Lee et al., 1997). By taking advantage of this property, we identify the neighboring objects by using only density information but without accessing the objects themselves or doing a lot of distance calculations. We further improve the efficiency by reducing the number of index nodes accessed using the *pruning* mechanism in a top-down search of the index.

Specifically, we first provide a formal definition of a cluster based on the notion of the density of regions (which we formally define in Section 3.2) in the multidimensional index. For this definition, we introduce the concept of the *region contrast partition*, which divides the database space into the higher-density section and the lower-density section based on the density of regions. Then, we present a branch-and-bound algorithm for pruning the index search to do the region contrast partition efficiently. Given two bounds (high and low), the pruning eliminates the index nodes whose region densities are out of the bounds. For this algorithm, we describe how the bounds are calculated and formally prove their correctness.

We demonstrate empirically that the proposed method is more efficient than BIRCH (Zhang et al., 1996), a well-known clustering algorithm, while producing clusters with the same or better accuracy. For this experiment, we use the elapsed time as the efficiency metric and introduce a new accuracy metric based on the relative number of objects in a cluster. The experimental results show that our algorithm is one or two orders of magnitude more efficient if we consider the index as already available from other applications. Even if we take the index creation and maintenance cost into consideration, our algorithm is significantly more efficient when the creation cost is amortized over a number of clustering operations performed until the index (if at all) needs to be recreated.

The rest of this paper is organized as follows. Section 2 introduces related work on existing clustering algorithms for large databases. Section 3 provides a formal definition of the cluster based on density represented by the multidimensional index. Section 4 presents the proposed top-down clustering algorithm. Section 5 shows the experimental results comparing the proposed algorithm and BIRCH. Finally, Section 6 summarizes and concludes the paper.

2. Related work

For an efficient clustering of large databases, some methods use sampling techniques (Breunig et al., 2001; Guha et al., 1998; Palmer and Faloutsos, 2000), and others use cluster summary information (Ganti et al., 1999; Zhang et al., 1996). The former methods extract samples from large databases and perform clustering on the samples. These methods have the advantage of being simple and easy to apply. However, they have the disadvantage that the accuracy of the clusters found depends largely on the sampling accuracy. The latter methods define the summary information that represents the shape of target clusters, and perform clustering using the information. These methods have the advantage of facilitating complex data analysis with a domain-specific summary. However, they have the disadvantage that the cluster shape is predetermined by the summary information.

BIRCH (Zhang et al., 1996) is a well-known clustering algorithm that uses summary information. It calculates the summaries of clusters (called clustering features (CFs)) from the original database, constructs a tree of nodes (called a CF-tree) containing the calculated summaries, and performs clustering using the tree instead of the original database. Each CF represents multiple objects. Thus, we can adjust the number of CFs by adjusting the number of objects represented by one CF. As a result, the size of the CF-tree can be adjusted depending on the available memory. This algorithm is fast because its clustering is based on CFs which are far fewer than the objects in the original database. Additionally, BIRCH is the first algorithm that handles noise objects (Sheikholeslami et al., 1998). However, BIRCH requires at least one scan over the entire database to build the initial CF-tree and does not yield accurate results for non-spherical clusters (Guha et al., 1998; Sheikholeslami et al., 1998).

CURE (Guha et al., 1998) is another clustering algorithm that uses summary information. Unlike BIRCH, which expresses a cluster with a single CF value, CURE selects several representative points for a cluster and calculates the distance between each of them and a point to check the membership of this point in the cluster. For scalability to large databases, it optionally performs sampling to reduce the search space. It further partitions the sampled space and then performs preclustering for each partition to obtain partial clusters, and finally merging them into clusters. Additionally, CURE offers a filtering function that prevents noise objects from being included in the clusters. However, CURE basically requires a scan over the entire database unless sampling is done. Even when sampling is used, since the result of clustering depends highly on the accuracy of sampling, it requires that samples of an adequate size must be used (Palmer and Faloutsos, 2000; Whang et al., 1990).

These methods using sampling or cluster summary have a critical drawback that the cluster quality deteriorates as we increase the data compression rate of sampling or summarizing. Recently, Breunig et al. (2001) proposed a method for increasing the data

compression rate using random sampling or summary (CF of BIRCH) without degrading the cluster quality. Palmer and Faloutsos (2000) proposed a density-biased sampling method, which solved the problem that uniform sampling is not suitable for finding relatively small-sized or low-density clusters.

WaveCluster (Sheikholeslami et al., 1998) uses a data transformation technique for clustering on large spatial databases. It performs clustering in a multidimensional data space mapped from the original database using wavelet transforms. Using multiresolution property of wavelet transforms, it can effectively identify clusters of arbitrary shapes at different levels of accuracy. Its efficiency, however, is affected significantly by the cost of accessing the entire database.

Regardless of the particulars of the techniques, all these methods require at least one exhaustive scan over the entire database *every time* clustering is performed. These limitations render the methods inapplicable to large databases.

3. Definition of the cluster

In this section we provide some definitions to formally define the cluster based on density represented by the multidimensional index. In Section 3.1, we introduce the terminology related to multidimensional indexes. In Section 3.2, we present the notion of the region contrast partition, which provides the basis of the proposed clustering method, and define a cluster based on the partition.

3.1. Terminology

Fig. 1 illustrates the structure of a multidimensional file and the names of its elements. The elements are categorized into *index pages*, which make the nodes of the index, and *data pages*, which store the data objects. An index page is either a leaf page or an internal page depending on its position in the hierarchy. A *leaf page* contains *leaf entries* made of < key, object identifier > pairs, and an *internal page* contains *internal entries* made of < key, child pointer > pairs. The root page is a special case of an internal page.

We call a region specified by an entry in the index page an *index region* or simply a *region*. Specifically, we call a region specified by a leaf entry a *leaf region* and a region specified by an internal entry an *internal region*. We define the *density of a region* as the ratio of the number of objects in the region over the size of the region. Note that in this paper we consider only rectangular regions, consistently with the references Kumar (1994), Seeger and Kriegel (1990) and Whang and Krishnamurthy (1985). Additionally, we consider only the indexes built using the *region-oriented splitting strategy* (Lee et al., 1997). This strategy always bisects a region so that two arbitrary regions do not overlap unless they are inclusive.

We now define notions of adjacency in a multidimensional index.

Definition 1. Consider two regions R_A and R_B in a k-dimensional space such that $R_A = [a_{x_1}, a_{y_1}] \times \cdots \times [a_{x_k}, a_{y_k}]$ and $R_B = [b_{x_1}, b_{y_1}] \times \cdots \times [b_{x_k}, b_{y_k}]$, where $[a_{x_i}, a_{y_i}]$ and $[b_{x_i}, b_{y_i}]$ for $i = 1, \ldots, k$ each denotes the interval on the ith dimensional axis. If only one dimension satisfies the following Condition 1 and the other $(k-1)$ dimensions satisfy the following Condition 2, then we say that the two regions R_A and R_B are *adjacent* and denote it as $R_A \oplus R_B$.

Condition 1 $(a_{x_i} = b_{y_i}) \vee (b_{x_i} = a_{y_i})$, $1 \leqslant i \leqslant k$
Condition 2 $(a_{x_j} \leqslant b_{x_j} < a_{y_j}) \vee (b_{x_j} \leqslant a_{x_j} < b_{y_j})$, $1 \leqslant j \leqslant k$, $j \neq i$

Definition 2. Given two regions R_A and R_B, if either $R_A \oplus R_B$ or there exists at least one sequence $\{R_A, R_1, R_2, \ldots, R_k, R_B\}$ such that $R_A \oplus R_1$, $R_1 \oplus R_2, \ldots, R_k \oplus R_B$, then we say that R_A and R_B are *transitively adjacent* and denote it as $R_A \otimes R_B$.

Example 1. Fig. 2 illustrates a two-dimensional index structure constructed using the region-oriented splitting strategy. The region A and the region B are adjacent (i.e.,

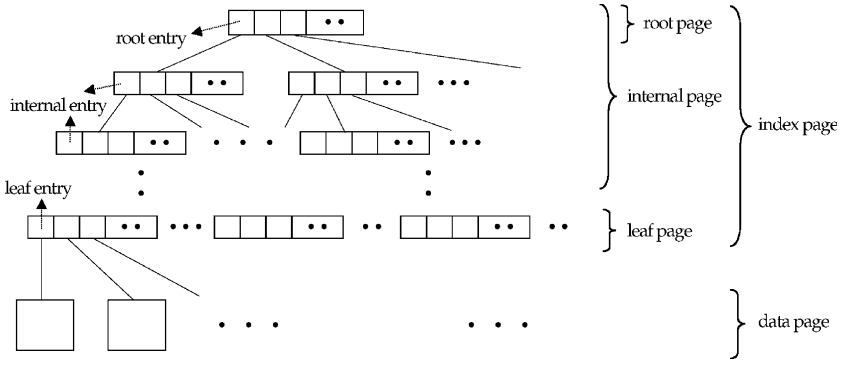

Fig. 1. Multidimensional file structure.

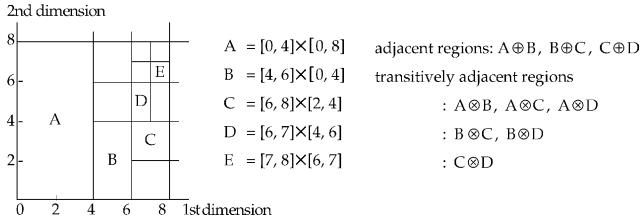

Fig. 2. Example of adjacency and transitive adjacency between regions in a two-dimensional index.

$A \oplus B$) by Definition 1. Specifically, the first dimensions of A and B satisfy Condition 1 and their second dimensions satisfy Condition 2. Likewise, the regions B and C and the regions C and D are adjacent as well. The regions B and D, A and C, and A and D respectively do not satisfy Condition 1, and therefore, are not adjacent. But, they are transitively adjacent. For example, there exists a sequence $\{A, B, C, D\}$ cascaded by the adjacent regions $A \oplus B$, $B \oplus C$, $C \oplus D$ between the regions A and D. Note that the region A and the region E are not transitively adjacent because no such sequence exists between them.

We define a clustering factor as the criterion for partitioning the entire index region into high object-density regions and low object-density regions.

Definition 3. A *clustering factor*, denoted by ρ, is defined as the ratio of the number of all objects in high object-density regions over the total number of objects stored in the database.

The clustering factor may be used to specify the minimum number of objects that must be included in high object-density regions relative to the total number of objects in the database. Additionally, it allows a certain number of noise objects to exist by excluding low object-density regions from the cluster. Those noise objects are called *removable objects*, and their number is denoted by NRO.

Table 1 summarizes the notations used in this paper.

Table 1
Summary of notations

Symbol	Definition/meaning
N	Total number of objects stored in database
k	Number of dimensions of a multidimensional index (cluster)
$s(R)$	Size of region R
$n(R)$	Number of objects included in region R
$d(R)$	Density of region $R(= n(R)/s(R))$
$R_i \oplus R_j$	Means that regions R_i and R_j are adjacent
$R_i \otimes R_j$	Means that regions R_i and R_j are transitively adjacent
ρ	Clustering factor provided by a user ($0 < \rho < 1$)
NRO	Number of removable objects ($= (1 - \rho) * N$)
ε	Cluster qualifier (see Condition 3 in Definition 5)

3.2. Region contrast partition and cluster definition

In this section we propose a method for partitioning a region using the density information and the clustering factor, and give a formal definition of a cluster based on the partition.

Definition 4. Given a region $\mathbb{R}$ consisting of a set of disjoint smaller regions $\{R_1, R_2, \ldots, R_m\}$ and NRO removable objects, the *region contrast partition* of $\mathbb{R}$ is defined as the pair of two sets, $\{\mathbb{R}_\mathbb{D}$ and $\mathbb{R}_\mathbb{S}\}$, that satisfy the following three conditions.

Condition 1 $\forall i, j((R_i \in \mathbb{R}_\mathbb{S}) \wedge (R_j \in \mathbb{R}_\mathbb{D}) \Rightarrow d(R_i) \leqslant d(R_j))$
Condition 2 $\sum_{R_i \in \mathbb{R}_\mathbb{S}} n(R_i) \leqslant \mathrm{NRO}$
Condition 3 $\sum_{R_i \in \mathbb{R}_\mathbb{S}} n(R_i) + n(R_p) > \mathrm{NRO}$, where R_p is the region with the lowest density in $\mathbb{R}_\mathbb{D}$

We call the regions included in $\mathbb{R}_\mathbb{D}$ and $\mathbb{R}_\mathbb{S}$ the *dense regions* and the *sparse regions*, respectively. If $\mathbb{R}$ consists of leaf regions, we call those included in $\mathbb{R}_\mathbb{D}$ the *dense leaf regions* and those included in $\mathbb{R}_\mathbb{S}$ the *sparse leaf regions*. In addition, we call R_p in Condition 3 the *partition boundary region*. From Definition 4, it is always possible to find a region contrast partition in a multi-dimensional index that uses the region-oriented splitting strategy.

Now we define the concepts of a cluster and clustering as used in the proposed method. The definitions are based on the set $\mathbb{R}_\mathbb{D}$ of dense regions obtained as a result of the region contrast partition.

Definition 5. Given a set of dense leaf regions $\mathbb{R}_\mathbb{D} = \{R_1, R_2, \ldots, R_p\}$, we define a *cluster* as the set $\mathbb{C}$ that satisfies the following three conditions.

Condition 1 $\mathbb{C} \subseteq \mathbb{R}_\mathbb{D}$
Condition 2 $(R_i \in \mathbb{C}) \wedge (R_i \otimes R_j) \Rightarrow R_j \in \mathbb{C}$
Condition 3 $\sum_{R_i \in \mathbb{C}} n(R_i) > \varepsilon, (\varepsilon \ll N)$

In this definition, Condition 2 requires that all dense leaf regions that are transitively adjacent should form a single cluster. Condition 3 requires that a single cluster should include at least ε objects. We call ε the *cluster qualifier*.

Definition 6. *Clustering* is the process of creating a region contrast partition for leaf regions to obtain a set $\mathbb{R}_\mathbb{D}$ of dense leaf regions and finding all clusters as defined in Definition 5 included in the set.

Now we can consider a basic density-based clustering algorithm based on the region contrast partition. We presented the basic algorithm and addressed that the algorithm is a direct realization of Definitions 1–6 in Hwang et al. (2003).

4. A top-down approach for density-based clustering

In this section we propose a top-down approach for density-based clustering that uses a multidimensional index. In Section 4.1, we introduce the concept of density-based pruning using internal entry information. In Section 4.2, we present an efficient top-down clustering algorithm using a branch-and-bound pruning mechanism.

4.1. Concept of density-based pruning

We now introduce the concept of density-based pruning for improving the clustering efficiency. The process of creating a region contrast partition for leaf regions requires all index pages of a multidimensional index be accessed to find the set of dense regions. This overhead can be alleviated using density-based pruning, which determines whether the leaf regions included in an internal region are all dense or all sparse using only the information stored in the internal entry (i.e., without accessing all leaf entries of the index) and prunes the search space accordingly. If all leaf regions within an internal region are dense, the internal region is called a *dense internal region*. In contrast, if all the leaf regions are sparse, the internal region is called a *sparse internal region*.

Example 2. Fig. 3 shows the leaf pages representing leaf regions and the corresponding internal pages. D denotes *dense*, and S denotes *sparse*. The leaf regions D_1, D_2, D_3, and D_4 of the internal entry D_0 are all dense, and S_1, S_2, S_3, and S_4 of S_0 are all sparse. Hence, D_0 is a dense internal region, and S_0 is a sparse internal region.

4.2. Density-pruning clustering algorithm

We now present an efficient clustering algorithm using density-based pruning. The algorithm employs a branch-and-bound mechanism for efficient pruning. It uses two kinds of density information—the highest density (d_highest) and the lowest density (d_lowest)—maintained in an internal entry of the multidimensional index. The value d_highest is the density of the leaf region with the highest density and the value d_lowest that of the leaf region with the lowest density among all the leaf regions included in the region represented by the internal entry. The algorithm creates a region contrast partition while searching the index in the breadth-first order deciding whether or not to search the lower level using the bounds. In this way, we can find dense (leaf or internal) regions of various sizes without accessing all leaf pages of the index. Finally, we find clusters from the set of these dense regions.

Fig. 4 shows the *density_pruning_clustering* algorithm. First, in line 2, it calculates the number of removable objects NRO using the clustering factor ρ and the number of objects in the database N. Second, in line 5, it accesses all internal pages at the current level of the multidimensional index *md*_index and constructs a list L of internal regions. Third, in lines 6 through 11, it sorts the list L in the ascending order of d_highest once and of d_lowest once to create the lists L_{high} and L_{low} respectively. Then, given $L_{\text{high}}(L_{\text{low}})$, it finds the partition boundary region $R_p^{\text{high}}(R_p^{\text{low}})$ and sets $B_{\text{high}}(B_{\text{low}})$ to the d_highest (d_lowest) of the region $R_p^{\text{high}}(R_p^{\text{low}})$. These B_{high} and B_{low} are the upper bound and the lower bound used to determine whether to prune or not in the next step. Fourth, in lines 12 through 19, it performs density-based pruning in the following two steps: (1) for each entry in L_{low} that satisfies $d\_\text{lowest} > B_{\text{high}}$, insert the corresponding region R_i^{low} into the set of dense regions *R_dense* and prune the subtree whose root node represents R_i^{low}, and (2) for each entry in L_{high} that satisfies $d\_\text{highest} < B_{\text{low}}$, decrease NRO by the number of objects included in the corresponding region R_i^{high} and prune the subtree whose root node represents R_i^{high}. If all removable objects have not been completely removed yet (i.e., NRO > 0 in line 4) as a result of searching the current level, it updates the current level to the next lower level (in line 20) and repeats the steps in lines 5 through 19. This iteration continues until either no removable object is left (i.e., NRO $= 0$) or the current level reaches the leaf region. If NRO > 0 even after the iteration is terminated, in lines 23 through 26, it constructs a list of the remaining leaf regions and creates a

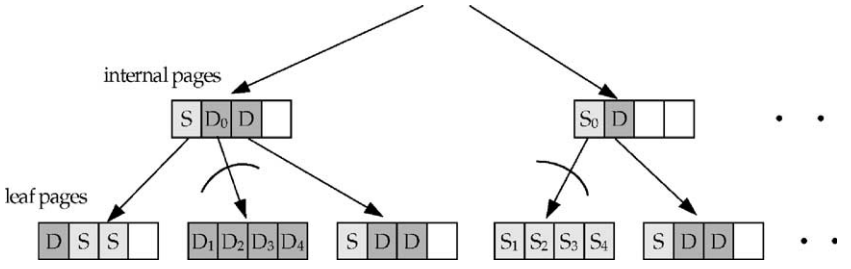

Fig. 3. Dense internal region and sparse internal region.

Algorithm *Density_Pruning_Clustering*
Input: *md_index* : multidimensional index to be used for clustering
N : the number of objects stored in the database
ρ : clustering factor
ε : cluster qualifier
Output: Set of Clusters
begin
1: *R_dense* := { };
2: *NRO* := (1-ρ)*N;
3: *curr_level* := root level of *md_index*;
4: **while** (*NRO* > 0 and *curr_level* < leaf level of *md_index*) **begin**
5: Construct a list L of internal regions by reading all pages at the current level from *md_index*;
6: Make a sorted list $L_{high}(=R_1^{high}, R_2^{high}, \ldots, R_m^{high})$ by sorting L in the ascending order of *d_highest*;
7: Find *partition boundary region* R_p^{high} from the sorted list L_{high} using *NRO*;
8: B_{high} := *d_highest*(R_p^{high});
9: Make a sorted list $L_{low}(=R_1^{low}, R_2^{low}, \ldots, R_m^{low})$ by sorting L in the ascending order of *d_lowest*;
10: Find *partition boundary region* R_p^{low} from the sorted list L_{low} using *NRO*;
11: B_{low} := *d_lowest*(R_p^{low});
12: **for** each R_i^{low} whose *d_lowest*$(R_i^{low}) > B_{high}$ **begin** /* R_i^{low} is a dense internal region */
13: *R_dense* := *R_dense* $\cup \{R_i^{low}\}$;
14: Prune the subtree whose root is the internal page representing R_i^{low};
15: **end**
16: **for** each R_i^{high} whose *d_highest*$(R_i^{high}) > B_{low}$ **begin** /* R_i^{high} is a sparse internal region */
17: *NRO* := *NRO* − $n(R_i^{high})$;
18: Prune the subtree whose root is the internal page representing R_i^{high};
19: **end**
20: *curr_level* := *curr_level* + 1;
21: **end**
22: if (*NRO* > 0) **begin** /* *curr_level* reached the leaf level */
23: Construct a list L of leaf regions by reading all remaining leaf pages from current *md_index*;
24: Make a sorted list L_{sort} $(=R_1, R_2, \ldots, R_m)$ by sorting L in the ascending order of region density;
25: Find *partition boundary region* R_p from the sorted list L_{sort} using *NRO*;
26: *R_dense* := *R_dense* $\cup \{R_p, R_{p+1}, \ldots, R_m\}$;
27: **end**
28: Find the set of clusters C from *R_dense* according to Definition 5;
29: Return C;
end

Fig. 4. The clustering algorithm based on density-based pruning.

region contrast partition to find the partition boundary region R_p using the current value of NRO. Then it adds the set of leaf regions with higher density than R_p to *R*_dense. Last, in lines 28 through 29, it performs clustering on those regions.

Theorem 1. *In density_pruning_clustering, a region whose* *d*_lowest *is greater than* B_{high} *is a dense internal region, and a region whose* *d*_highest *is smaller than* B_{low} *is a sparse internal region.*

Proof. It suffices to prove only the first part (regarding a dense internal region) because the second part is symmetrical. We prove it by proof-by-contradiction. Let $\mathbb{R}^{\text{I}} = \{R_1^{\text{I}}, \ldots, R_x^{\text{I}}\}$ be the set of internal regions with lower densities than B_{high} among the internal regions in the list L_{high}, and $\mathbb{R}^{\text{L}} = \{R_1^{\text{L}}, \ldots, R_y^{\text{L}}\}$ be all leaf regions included in $\mathbb{R}^{\text{I}}$. If we let $\mathbb{R}_{\mathbb{S}}$ be the set of sparse leaf regions and $\mathbb{R}_{\mathbb{D}}$ the set of dense leaf regions, then there always exists R_i^{L} in $\mathbb{R}_{\mathbb{D}}$ $(i = 1, \cdots, y)$ such that $R_i^{\text{L}} \in \mathbb{R}^{\text{L}} - \mathbb{R}_{\mathbb{S}}$ because $n(\mathbb{R}^{\text{I}}) = n(\mathbb{R}^{\text{L}}) >$ NRO by the definition of B_{high}. Next, let S^{I} be an arbitrary internal region where *d*_lowest $> B_{\text{high}}$ in the list L_{low}, and let $\mathbb{S}^{\text{I}} = \{S_1^{\text{L}}, \ldots, S_z^{\text{L}}\}$ be the set of all leaf regions included in S^{I}.

Assumption part: Let us assume that an arbitrary region S_j^{L} in $\mathbb{S}^{\text{I}}$ is a sparse leaf region

(i.e., $S_j^L \in \mathbb{R}_S$). Then $d(R_i^L) > d(S_j^L)$ follows from $S_j^L \in \mathbb{R}_S$ and $R_i^L \in \mathbb{R}_D$.

Contradiction part: From the definitions of B_{high} and d_lowest, $d(R_i^L) \leqslant B_{high}$ holds true for the region R_i^L in $\mathbb{R}^L$, and $d(S_j^L) \geqslant d\_lowest$ holds for the region S_j^L in $\mathbb{S}^I$. Then, because $d\_lowest > B_{high}$ for region S^I (by definition), $d(R_i^L) < d(S_j^L)$ also holds. However, this contradicts the assumption $d(R_i^L) > d(S_j^L)$. Therefore, we conclude that S_j^L is a dense leaf region and $\mathbb{S}^I$ is a dense internal region, which consists of only dense leaf regions. □

Example 3. Fig. 5 illustrates the steps of executing the density-based pruning algorithm. Suppose the internal regions of an index at the current level are as shown in Fig. 5(a), and each region has the information shown in Fig. 5(b). Fig. 5(d) is a list of the regions sorted by *d_highest* and shows how the upper bound B_{high} is determined if $\rho = 0.9$. Fig. 5(e) shows a list of the regions sorted by *d_lowest* and shows how the lower bound B_{low} is determined. We see from Fig. 5(d) and 5(e) that *d_highest* is smaller than B_{low} for the region R_1. This shows that all leaf regions included in the region R_1 are sparse leaf regions. Therefore, the subtree with the region R_1 as the root may be pruned. On the other hand, the regions R_4 and R_8 have *d_lowest* higher than B_{high}. That is, all leaf regions included in the two regions are dense leaf regions. Therefore, the subtrees with the two regions as their roots may be pruned. Fig. 5(c) shows the regions after the pruning.

Corollary 1. *Clusters obtained by density_pruning_clustering are identical to the clusters defined in Definition* 5.

Proof. It is straightforward from the fact that density-based pruning is a branch-and-bound algorithm and the correctness of its bounds has been proven in Theorem 1. □

5. Performance evaluation

In this section we present the results of comparing our proposed algorithm *density_pruning_clustering* with BIRCH (Zhang et al., 1996), a widely-known clustering algorithm. First, we describe the experimental data and environment in Section 5.1. Then, we evaluate the efficiency of obtaining clusters in Section 5.2, the accuracy of clusters in Section 5.3, and the sensitivity of cluster accuracy to the clustering factor in Section 5.4.

5.1. Experimental data and environment

We perform experiments using three types of synthetic data sets depicted in Fig. 6. Each data set consists of two-dimensional point objects, and each dimension is an integer in the domain $[-2^{20}, 2^{20}-1]$. We generate clusters using the normal distribution and generate noise objects using the uniform distribution. Here, we adjust the sizes and shapes of clusters by changing the standard deviation (σ) of the normal distribution and the correlation coefficient (ξ) between the two dimensions. We use a standard deviation ranging between 2^{14} and 2^{16}.

We generate each data set shown in Fig. 6 with the following specifications.

- DS1: This set consists of five clusters of various shapes. For the three clusters of spherical and elliptical shapes, we use $\sigma = 2^{15}$ and $\xi = 0$, 0.5, and −0.5. For the cluster consisting of two connected ellipses, we use $\sigma = 2^{16}$ and $\xi = 0.8$ and −0.8. For the donut-shaped cluster, we first generate a spherical cluster with $\sigma = 2^{16}$ and $\xi = 0$, and then remove the objects that are the nearest 30% and the farthest 15% from the center of the cluster.
- DS2: This set consists of 25 spherical clusters, and each cluster contains the same number of objects.

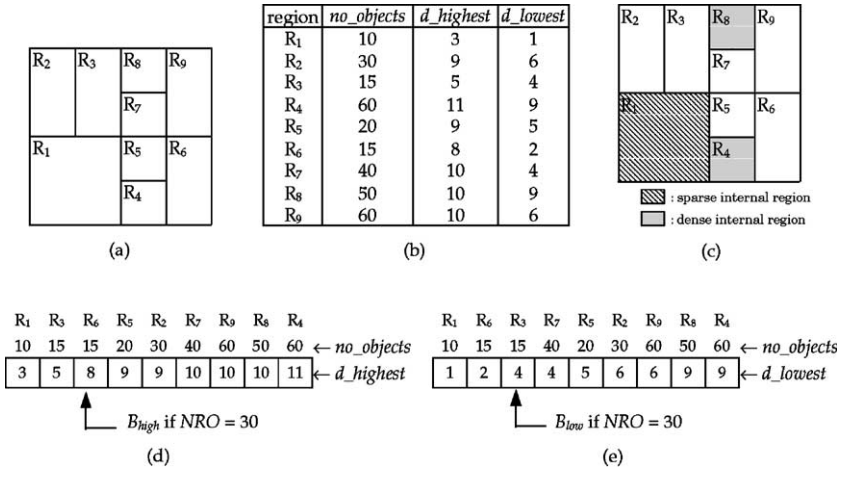

Fig. 5. An example of executing the density-based pruning algorithm. (a) Internal regions of index, (b) index entry informations of regions, (c) result of pruning, (d) determination of upper bound, (e) determination of lower bound.

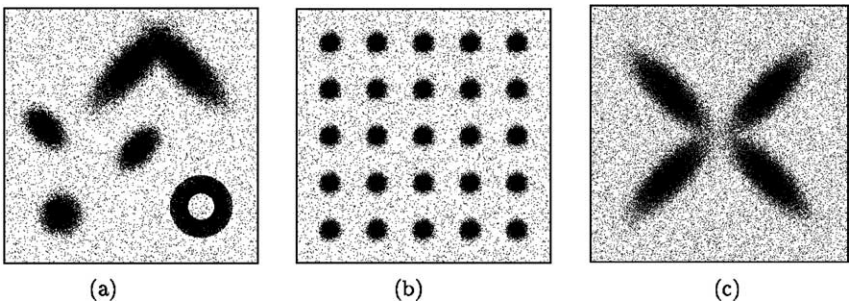

Fig. 6. Test data sets. (a) DS1, (b) DS2 and (c) DS3.

We generate each cluster with $\sigma = 2^{14}$ and $\xi = 0$. We arrange these 25 clusters as a regular grid in the two-dimensional space.

- DS3: This set consists of four elliptical clusters forming the shape of a cross. To generate each cluster, we use $\sigma = 2^{16}$ and $\xi = 0.8$ and -0.8.

DS1 is characterized by the variety of the shapes and sizes of the clusters, and DS2 by its similarity to the data set used in BIRCH. We use DS1 and DS2 to evaluate the cluster accuracy and the clustering efficiency, and DS3 to analyze the sensitivity of clustering to the clustering factor. Further details of the data sets, such as the number of objects in each data set, the noise ratio in each data set, and the number of objects in each cluster, are also varied in each experiment.

All the experiments are conducted on a SUN Ultra 60 workstation with 512MB of main memory. We use a multilevel grid file (MLGF) (Lee et al., 1997; Whang and Krishnamurthy, 1985) as the multidimensional file structure for storing data, and set the data page size and the index page size equally to 1024 bytes. [1]

The MLGF is a dynamic hierarchical balanced multidimensional file structure for point data. It consists of a multilevel index and data pages, and employs the region-oriented splitting strategy.

5.2. Evaluation of clustering efficiency

In this subsection we show the results of comparing the efficiency of the two algorithms: *density_pruning_clustering* (DP) and BIRCH. We use the elapsed time as the metric of efficiency.

Here, we compare our method with BIRCH, which is a representative clustering algorithm. The comparison with other existing algorithms (CURE, WaveCluster) can be done through relative comparison of these algorithms with BIRCH that have been presented in the literature (Guha et al., 1998; Sheikholeslami et al., 1998). It has been reported that CURE (Guha et al., 1998) is two to five times better than BIRCH and WaveCluster (Sheikholeslami et al., 1998) is eight to ten times better than BIRCH with data sets each consisting of some 100,000 objects.

We generate three data sets of different sizes for DS1 and DS2, respectively. These data sets contain one hundred thousand, one million, and ten million objects, respectively. We set the noise ratio to 8% of the total number of objects, which is equivalent to setting the clustering factor ρ to 0.92. In addition, we set the cluster qualifier ε to 0.1% of the total number of objects. In executing BIRCH, we use the default values recommended by Zhang et al. (1996) as the input parameters.

Fig. 7 compares the elapsed time between DP and BIRCH for the data set DS1. BIRCH constructs the memory-based *CF-tree* as the first step (*Phase* 1) of the clustering process. To compare the performance on a fair basis when the multidimensional index (MLGF) already exists, we also present the elapsed time for BIRCH without phase 1. The result shows that DP reduces the time by 21 to 96 times compared with BIRCH (without Phase 1). This is due to the effect of density-based pruning. Note that DP's performance gain increases as the number of objects increases. The reason is that having more objects (and accordingly more pages) leads to a finer partitioning of the space and increases the number of dense or sparse internal regions, thus pruning more search space.

Table 2 shows the quantitative results of the density-based pruning performed on DS1 by DP. We see that DP prunes 44.2% to 91.4% of all index pages and 55.7% to 93.6% of all objects.

Fig. 8 shows the elapsed time with respect to the size of DS2. The results are similar to those in Fig. 7. DP reduces the time by 15–87 times compared with BIRCH (without Phase 1). The percentages of pruned pages and pruned objects are in the ranges of 25.6–89.3% and 49.3–92.3%, respectively.

If there is no existing multidimensional index available for clustering, the clustering cost may well include the cost of creating an index. In addition, the cost should include the cost of maintaining the index for updates. Table 3 shows the resulting elapsed time of DP in com-

[1] The reason for using such a relatively small page size is to make the multidimensional index high enough to check the pruning efficiency. Similar results would be obtained with the page size of 4096.

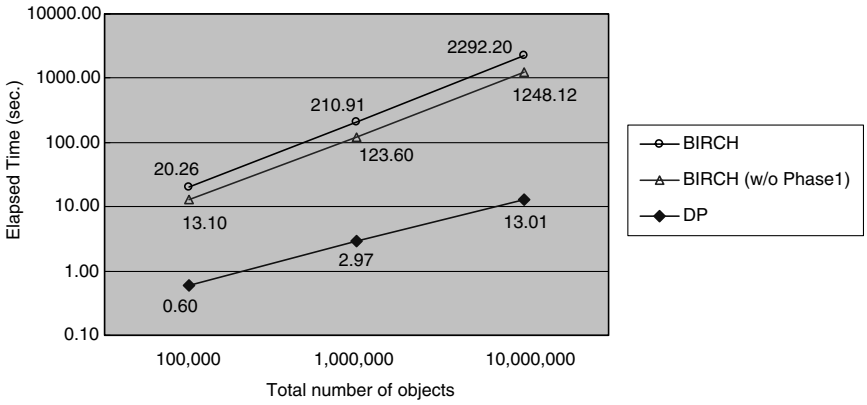

Fig. 7. Elapsed time with respect to the size of DS1 for each algorithm.

Table 2
The results of density-based pruning by DP for three sizes of DS1

Total number of objects	Total number of index pages	Number of index pages pruned (%)	Number of objects pruned (%)
100,000	276	122 (44.2)	55,695 (55.7)
1,000,000	2544	1865 (73.3)	803,305 (80.3)
10,000,000	24,834	22,698 (91.4)	9,361,793 (93.6)

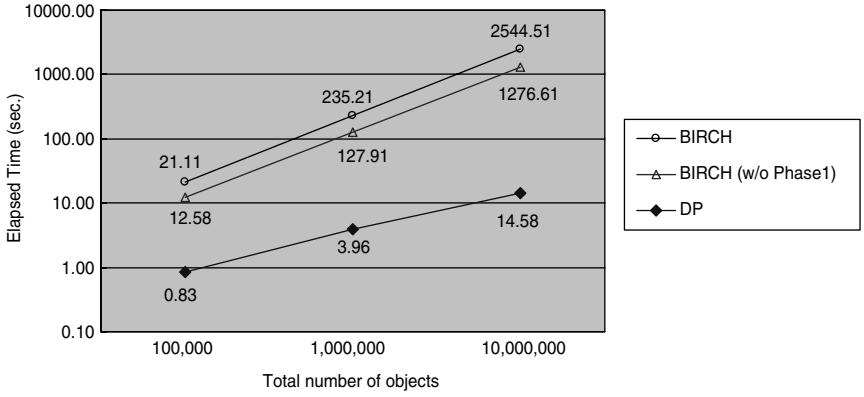

Fig. 8. Elapsed time with respect to the size of DS2 for each algorithm.

Table 3
Elapsed time in seconds of BIRCH and DP considering the index creation and maintenance overheads

Methods	Data set DS1 (million objects)			Data set DS2 (million objects)		
	0.1	1	10	0.1	1	10
BIRCH	20.26	210.91	2292.20	21.11	235.21	2544.51
BIRCH (w/o Phase 1)	13.10	123.60	1248.12	12.58	127.91	1276.61
BIRCH index creation (CF-tree)	7.16	87.31	1044.08	8.53	107.30	1267.90
DP(100)	0.92	8.26	174.81	1.17	9.13	169.26
DP(1000)	0.66	3.78	33.23	0.89	4.76	33.92
DP(∞)	0.63	3.28	17.50	0.86	4.28	18.89
DP	0.60	2.97	13.01	0.83	3.96	14.58
Index creation (MLGF)	29.65	498.43	15,731.71	30.71	485.56	15,037.35
Index maintenance (MLGF)	0.02	0.31	4.48	0.03	0.32	4.30

parison with that of BIRCH. Index creation time has been amortized by diving it by the number of repeated clustering until the time the index (if at all) needs to be recreated, and the index maintenance time reflects in-

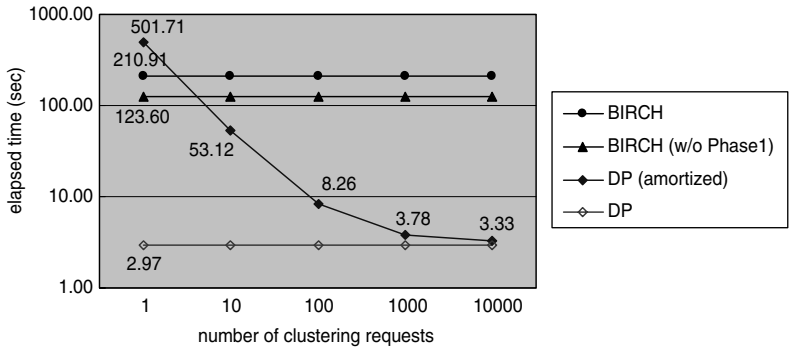

Fig. 9. Elapsed time of DP and BIRCH with respect to the number of clustering requests processed with DS1 (1 million objects), where DP (amortized) indicates DP including the amortized index creation time and the maintenance time.

serting or deleting 0.1% [2] of all the objects in the database between two consecutive clustering operations.

In Table 3, DP(n) denotes the cost of DP when the index creation cost is amortized over n clustering operations. We have used 100, 1000 and ∞ as the value of n. DP(∞) denotes the cost amortized over the number of clustering operations large enough to erase the upfront index creation cost, thus considering only the maintenance cost. The table shows that DP is significantly more efficient than BIRCH even after considering the overhead of index creation and maintenance.

Fig. 9 shows the resulting amortized elapsed time of DP with DS1 (1 million objects) in comparison with BIRCH. It shows how fast the cost is amortized over repeated executions of clustering, and also shows that DP is significantly (by at least an order of magnitude when the number of clustering requests $\geqslant 100$) more efficient than BIRCH even after considering the overhead of index creation and maintenance.

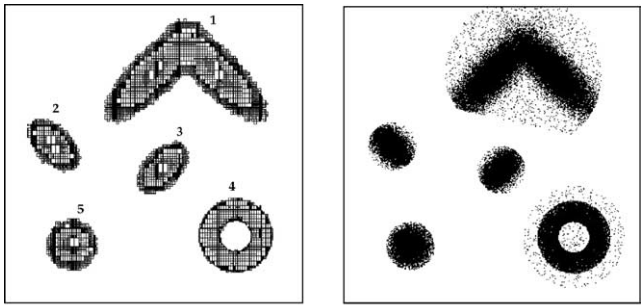

Fig. 10. Clustering results for DS1 by each algorithm. (a) DP, (b) BIRCH.

We define the following metric to measure the accuracy of clusters. [3]

$$\text{accuracy of cluster } \mathbb{C} = \frac{\text{the number of objects composing the main (normal) distribution in cluster } \mathbb{C} \text{ (Main)}}{\text{the total number of objects in cluster } \mathbb{C} \text{ (Total)}}$$

5.3. Evaluation of cluster accuracy

In this subsection we compare the accuracy of the clusters found by DP and BIRCH. We generate one data set for both DS1 and DS2 so that each data set contains one million objects. As in Section 5.2, we set the noise ratio to 8% of the total number of objects, equivalently, setting the clustering factor ρ to 0.92; set the cluster qualifier ε to 0.1%; and use the default values recommended by Zhang et al. (1996) as the input parameters to execute BIRCH.

Fig. 10 shows the clusters found from DS1 with the two algorithms. As shown in Fig. 10(a), the clusters found by DP are very similar in shape to those in the original data set shown in Fig. 6(a). This similarity strongly indicates that DP is an accurate method. We can see in the figure that the insides of clusters consist of many small or big rectangles. This is because that the density-based pruning occurs at many internal regions and these regions are larger than the leaf regions. The result of BIRCH in Fig. 10(b) shows that all the clusters found have spherical shapes. The reason is that BIRCH constructs a tree with clustering features (CFs) each consisting of the center value and radius, naturally representing a cluster of a spherical shape.

[2] This would be an upper bound of the update rate in most practical applications except for the case of unusually heavy updates.

[3] The total number of objects in cluster $\mathbb{C}$ includes noise objects created by using the uniform distribution.

Table 4
Accuracy of the clusters found from DS1 by each algorithm

Clusters	Number of objects				Cluster accuracy (%)	
	DP		BIRCH		DP	BIRCH
	Total[a]	Main[b]	Total	Main		
Cluster 1	398,638	390,935	404,076	388,644	98.1	96.2
Cluster 2	99,459	97,842	98,112	96,344	98.4	98.2
Cluster 3	99,517	97,862	97,608	95,913	98.3	98.3
Cluster 4	222,800	219,631	227,364	220,000	98.6	96.8
Cluster 5	99,402	97,556	97,457	95,904	98.1	98.4

[a] Total: The number of total objects included in cluster $\mathbb{C}$.
[b] Main: The number of objects composing the main distribution in cluster $\mathbb{C}$.

Table 4 shows the accuracy of the clusters found from DS1. As we see in the table, the proposed algorithms show uniformly high accuracy ranging 98.1–98.6% for all clusters. The accuracy of BIRCH is also high in the range of 96.2–98.4% because a majority of dense regions are included in the spherical clusters found. In particular, we see that the accuracy of BIRCH is even slightly higher than DP when the shape of a cluster is exactly spherical. This effect is seen because BIRCH is particularly tuned for finding spherical clusters.

Fig. 11(a) through Fig. 11(b) shows the clusters found from DS2 when $\rho = 0.92$ and $\varepsilon = 0.001$. As we can see in Fig. 11(a), the clusters found by DP are very similar in shape to those of the original data set in Fig. 6(b).

This indicates that DP finds clusters accurately even when there are a large number of clusters. It also shows that density-based pruning occurs significantly as it does with DS1. As we can see in the spherical clusters of Fig. 11(b), BIRCH also find clusters accurately. The accuracy of clusters from DS2 is in the range of 98.5–98.7% for DP and 98.6–99.0% for BIRCH. BIRCH appears slightly better here because the clusters are exactly spherical. We note that BIRCH always finds spherical-shaped clusters while DP finds arbitrary ones.

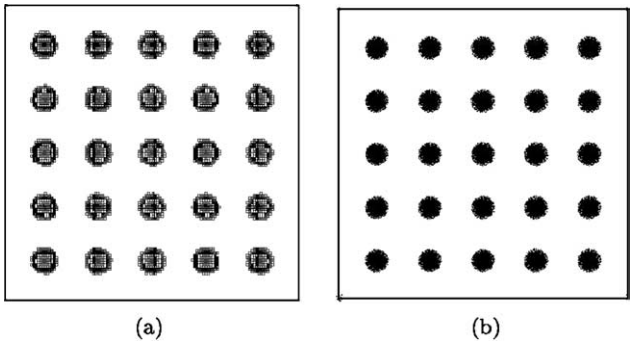

Fig. 11. Clustering results for DS2 by each algorithm. (a) DP, (b) BIRCH.

5.4. Sensitivity of cluster accuracy to the clustering factor

In this subsection we analyze the clustering results while changing the clustering factor ρ or, equivalently, by changing the noise level of the data set. (Note that the noise level equals $(1 - \rho) \times 100\%$.) We generate one data set of DS3 that contains 100,000 objects, of which 20% are noise objects, and sets the cluster qualifier ε to 0.1% of the total number of objects.

Fig. 12 shows the clustering results for three different values of ρ. Fig. 12(a) shows the clusters found when ρ is 0.7, that is, when 30% of data objects are noise objects. We see that total four clusters are found as a result of eliminating sparse regions. Fig. 12(b) shows the clusters found when ρ is 0.8. Among the regions eliminated in Fig. 12(a), some regions of relatively high density are labeled dense regions and become transitively adjacent. As a result, only two clusters are found. Fig. 12(c) shows the clustering result when ρ is 0.85. More regions are labeled dense regions, and all dense regions become transitively adjacent. As a result, we find only one cluster.

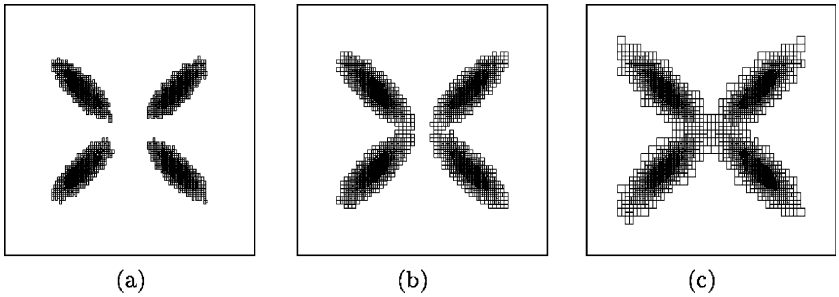

Fig. 12. Result of sensitivity analysis for different clustering factors (or noise levels). (a) $\rho = 0.70$, (b) $\rho = 0.80$, (c) $\rho = 0.85$.

6. Conclusions

In this paper, we have proposed a novel top-down clustering method based on region density using a multidimensional index. Generally, multidimensional indexes have inherent clustering property of storing similar (i.e., close to each other) objects in the same or adjacent data pages. By taking advantage of this property, our method finds similar objects using only the region density information without incurring the high cost of accessing the objects themselves and calculating distances among them.

First, we have provided a formal definition of the cluster based on the concept of region contrast partition. A cluster that we define is a set of dense regions that are adjacent to one another. The dense regions are identified by creating a region contrast partition.

Next, we have proposed the *density_pruning_clustering* (DP) algorithm. DP employs a branch-and-bound mechanism that improves efficiency by pruning unnecessary search in finding the set of dense regions. For this algorithm, we have presented the method for determining the bounds, B_{high} and B_{low}, and have formally proved the correctness of the bounds in Theorem 1.

To evaluate the performance of the proposed algorithm, we have conducted extensive experiments. Experimental results show that the accuracy of the proposed algorithm is similar or superior to that of BIRCH except for exactly spherical clusters. The results also show that the efficiency of the proposed algorithm is far superior to that of BIRCH due to density-based pruning. Experimental results for large data sets consisting of 10 million objects show that DP reduces the elapsed time by up to 96 times compared with that of BIRCH. Even with the cost of index creation and maintenance considered, the proposed algorithm is significantly (by an order of magnitude) more efficient than BIRCH. Further, we note that the improvement in performance becomes more marked as the size of the database increases, making this method more suitable for larger databases.

The top-down clustering approach proposed in this paper greatly improves the clustering performance for large databases without sacrificing accuracy. We believe that the proposed methods will be practically usable in many large database applications.

Acknowledgement

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc).

References

Ankerst, M., Breunig, M., Kriegel, H.P., Sander, J., 1999. OPTICS: Ordering points to identify the clustering structure. In: Proc. Int'l Conf. on Management of Data, ACM SIGMOD, pp. 49–60.

Breunig, M., Kriegel, H.P., Kroger, P., Sander, J., 2001. Data bubbles: quality preserving performance boosting for hierarchical clustering. In: Proc. Int'l Conf. on Management of Data, ACM SIGMOD, pp. 79–90.

Chen, M.S., Han, J., Yu, P.S., 1996. Data mining: an overview from a database perspective. IEEE Trans. Knowledge Data Eng. 8 (6), 866–883.

Ester, M., Kriegel, H.P., Xu, X., 1995. Knowledge discovery in large spatial databases: focusing techniques for efficient class identification. In: Proc. the Fourth Int'l Symp. on Large Spatial Databases (SSD), pp. 67–82.

Ester, M., Kriegel, H.P., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. the Second Int'l Conf. on Knowledge Discovery and Data Mining (KDD), pp. 226–231.

Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A., French, J., 1999. Clustering large datasets in arbitrary metric spaces. In: Proc. the 15th Int'l Conf. on Data Engineering (ICDE), pp. 502–511.

Guha, S., Rastogi, R., Shim, K.S., 1998. CURE: an efficient clustering algorithm for large databases. In: Proc. Int'l Conf. on Management of Data, ACM SIGMOD, pp. 73–84.

Hwang, J.J., Whang, K.Y., Moon, Y.S., Lee, B.S., 2003. Top-down clustering using multidimensional indexes, KAIST Technical Report CS-TR-2003-189. Available from <http://cs.kaist.ac.kr/new/english/rnd/public/tech2003.html>.

Karypis, G., Han, E.H., Kumar, V., 1999. Chameleon: hierarchical clustering using dynamic modeling. IEEE Comp. 32 (8), 68–75.

Kaufman, L., Rousseeuw, P.J., 1990. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons.

Kumar, A., 1994. G-tree: a new data structure for organizing multidimensional data. IEEE Trans. Knowledge Data Eng. 6 (2), 341–347.

Lee, J.H., Lee, Y.K., Whang, K.Y., Song, I.Y., 1997. A region splitting strategy for physical database design of multidimensional file organizations. In: Proc. the 23rd Int'l Conf. on Very Large Data Bases, pp. 416–425.

Ng, R.T., Han, J., 1994. Efficient and effective clustering methods for spatial data mining. In: Proc. the 20th Int'l Conf. on Very Large Data Bases, pp. 144–155.

Palmer, C.R., Faloutsos, C., 2000. Density biased sampling: an improved method for data mining and clustering. In: Proc. Int'l Conf. on Management of Data, ACM SIGMOD, pp. 82–92.

Schikuta, E., 1996. Grid-clustering: an efficient hierarchical clustering method for very large data sets. In: Proc. the 13th Int. Conf. on Pattern Recognition, vol. 2, pp. 101–105.

Sheikholeslami, G., Chatterjee, S., Zhang, A., 1998. WaveCluster: a multi-resolution clustering approach for very large spatial databases. In: Proc. the 24th Int'l Conf. on Very Large Data Bases, pp. 428–439.

Seeger, B., Kriegel, H.P., 1990. The buddy-tree: an efficient and robust access method for spatial data base systems. In: Proc. the 16th Int'l Conf. on Very Large Data Bases, pp. 590–601.

Whang, K.Y., Krishnamurthy, R., 1985. Multilevel Grid Files, IBM Research Report RC11516.

Whang, K.Y., Zanden, B.T.V., Taylor, H.M., 1990. A linear-time probabilistic counting algorithm for database applications. ACM Trans. Database Systems 15 (2), 208–229.

Zhang, T., Ramakrishnan, R., Livny, M., 1996. BIRCH: an efficient data clustering method for very large databases. In: Proc. Int'l Conf. on Management of Data, ACM SIGMOD, pp. 103–114.