ELSEVIER

Information Processing Letters 80 (2001) 87–95

Information Processing Letters

www.elsevier.com/locate/ipl

On the effective clustering of multidimensional data sequences

Seok-Lyong Lee [a], Chin-Wan Chung [b,*]

[a] *Department of Information and Communication Engineering, Korea Advanced Institute of Science and Technology, 373-1, Kusong-Dong, Yusong-Gu, Taejon 305-701, South Korea*

[b] *Department of Computer Science, Korea Advanced Institute of Science and Technology, 373-1, Kusong-Dong, Yusong-Gu, Taejon 305-701, South Korea*

Received 22 June 2000; received in revised form 25 October 2000
Communicated by F.Y.L. Chin

Abstract

In this paper, we investigate the problem of clustering multidimensional data sequences such as video streams. Each sequence is represented by a small number of hyper-rectangular clusters for subsequent indexing and similarity search processing. We present a linear clustering algorithm that guarantees the predefined level of clustering quality, and show its effectiveness via experiments on various video data sets.

Keywords: Algorithms; Clustering algorithms; Databases; Multidimensional data sequences

1. Introduction

Given a set of data points in a multidimensional space, the task of clustering is the grouping of the points into *clusters* such that points within each cluster have similar characteristics while those in different clusters are dissimilar. A data point that is considerably dissimilar to or inconsistent with the remainder of the data is referred to as an *outlier* or a *noise*. The problem we address in this paper is to design an effective scheme for clustering points in a multidimensional data sequence (MDS) for the subsequent indexing and similarity search. In [7], we have formally defined an MDS S in the n-dimensional space as a series of its component vectors, $S = \langle S[1], S[2], \ldots, S[k] \rangle$, where each vector $S[t]$ $(1 \leqslant t \leqslant k)$ is composed of n scalar entries, that is, $S[t] = (S^1[t], S^2[t], \ldots, S^n[t])$. A typical example of an MDS is a video stream. By representing each frame with a feature vector, depending on its attributes such as color, texture or shape, the video is modeled as a trail of points in the feature space such that each frame of the video constitutes a multidimensional point.

Various clustering methods have been studied in database communities, such as [1,2,4,8,9]. However, the clustering of MDS should be handled in a way different from traditional clustering methods in various aspects. Each object in existing methods is mapped into a single point and thus belongs to a single cluster, while an MDS is composed of multiple points that are contained in multiple separate clusters. The shapes of clusters may also be different. Traditional methods attempt to look at quantitative properties of clusters such as the mean square error, independent of how they will be used. Therefore, the shapes of clusters tend to be determined arbitrarily based on the distribution of

* Corresponding author.
E-mail addresses: sllee@islab.kaist.ac.kr (S.-L. Lee), chungcw@cs.kaist.ac.kr (C.-W. Chung).

0020-0190/01/$ – see front matter
PII: S0020-0190(01)00144-2

points in the space. On the other hand, we perform the clustering of MDS as a prior stage for the subsequent similarity search, such as '*Find video streams that are similar to a given stream of a news video*'. It means the shape of each cluster should be appropriate for the indexing mechanism. It is beneficial to consider a hyper-rectangle as a shape of a cluster, since currently dominant indexing mechanisms such as the R-tree [5] and its variants are based on a minimum bounding rectangle (MBR) as their node shape and thus we can use them without modification.

The similarity search applications of MDS place various special requirements on clustering techniques, motivating the need for designing a new clustering algorithm. Those requirements include

(1) minimizing the volume per point (VPP) of clusters,
(2) minimizing the edge (i.e., edge length) per point (EPP) of clusters,
(3) maximizing the number of points per cluster (PPC),
(4) dealing with outliers appropriately, and
(5) minimizing the number of input parameters.

The first three requirements imply that our method aims at generating dense clusters with respect to volume and edge since they have a great impact on the retrieval efficiency of large data sets. Clearly, denser clusters occupy smaller space. When clusters are indexed in the search space, if clusters are small, they have less possibility to intersect with the query region. Therefore, denser clusters have higher search efficiency than sparser clusters. This is a motivation of devising our clustering method. In this paper, the clustering problem is formalized as follows:

Given: A set of MDSs and the minimum number of points *minPts* per cluster.

Goal: To find a set of clusters C and a set of outliers O that optimize the values of specific measurement criteria.

An input parameter *minPts* is needed to judge outliers. More details on how to determine outliers are given in Section 3.3. One thing we need to clarify is that, even though our method is designed to run on the high (say, 50 or 70) dimensionality as well as the low (say, 3 or 5) dimensionality, its application has the restriction for the high dimensionality. For the retrieval of large data sets, high-dimensional data may not be used in reality since it causes severe processing overhead. Therefore, if the raw data is high-dimensional, it is usual that its dimensionality is reduced to avoid the '*dimensionality curse problem*'. In this context, our method fits well with the low dimensionality.

2. Clustering characteristics of MDS

2.1. Cluster representation

A hyper-rectangular cluster CL with k points, P_t for $t = 1, 2, \ldots, k$ in the normalized unit space $[0, 1]^n$, is represented by two endpoints, L (low point) and H (high point) of its major diagonal, and the number of points in the cluster as follows: $CL = \langle L, H, k\rangle$, where

$$L = \left\{(L^1, L^2, \ldots, L^n) \mid L^i = \min_{1\leqslant t\leqslant k}(P_t^i)\right\},$$

and

$$H = \left\{(H^1, H^2, \ldots, H^n) \mid H^i = \max_{1\leqslant t\leqslant k}(P_t^i)\right\}$$

for $i = 1, 2, \ldots, n$.

It is sometimes convenient if we represent a point P_t in the cluster form by placing $L = H = P_t$ and $k = 1$, that is, $\langle P_t, P_t, 1\rangle$. This cluster is denoted by $CL(P_t)$. The volume $Vol(CL)$ and the total edge length $Edge(CL)$ of CL are computed as:

$$\begin{aligned} Vol(CL) &= \prod_{1\leqslant i\leqslant n}(H^i - L^i), \\ Edge(CL) &= 2^{n-1}\cdot\sum_{1\leqslant i\leqslant n}(H^i - L^i). \end{aligned} \tag{1}$$

Then, the volume and edge of CL per point, $VPP(CL)$ and $EPP(CL)$ respectively, will be:

$$\begin{aligned} VPP(CL) &= \frac{Vol(CL)}{k} = \frac{\prod_{1\leqslant i\leqslant n}(H^i - L^i)}{k}, \\ EPP(CL) &= \frac{Edge(CL)}{k} = \frac{2^{n-1}\cdot\sum_{1\leqslant i\leqslant n}(H^i - L^i)}{k}. \end{aligned} \tag{2}$$

2.2. Operation between clusters

The clustering process generates larger clusters gradually by merging two clusters. We define the merging operator to do it as follows:

Definition 1 (*Merging operator* $\oplus$). Let CL_1 and CL_2 be clusters. For $CL_1 = \langle L_1, H_1, k_1\rangle$ and $CL_2 = \langle L_2, H_2, k_2\rangle$, the merging operator $\oplus$ is defined as $CL_1 \oplus CL_2 = CL_3$ such that

$$CL_3 = \langle L_3, H_3, k_3\rangle,$$
$$L_3 = \left\{(L_3^1, L_3^2, \ldots, L_3^n) \mid L_3^i = \min(L_1^i, L_2^i)\right\},$$
$$H_3 = \left\{(H_3^1, H_3^2, \ldots, H_3^n) \mid H_3^i = \max(H_1^i, H_2^i)\right\}$$
for $i = 1, 2, \ldots, n$, and $k_3 = k_1 + k_2$.

Suppose a cluster $CL_t = \langle L_t, H_t, k_t\rangle$ to be merged to a cluster $CL_s = \langle L_s, H_s, k_s\rangle$ in the unit space $[0, 1]^n$ during the clustering process, where each of CL_t and CL_s can be a cluster or a point. Merging two clusters produces a new, probably bigger, cluster, which causes changes in the volume, the edge, and the number of points. We are interested in the amount of change resulting from the merging process, since it is an important factor for clustering. (We address it later.) The volume and edge increments, $\Delta Vol(CL_s, CL_t)$ and $\Delta Edge(CL_s, CL_t)$, respectively, are formulated as follows:

$$\Delta Vol(CL_s, CL_t) = Vol(CL_s \oplus CL_t) - Vol(CL_s), \quad (3.1)$$

$$\Delta Edge(CL_s, CL_t) = Edge(CL_s \oplus CL_t) - Edge(CL_s). \quad (3.2)$$

Using above formulae, we can get the mean increment of volume and edge per point, ΔVPP and ΔEPP, respectively, in the cluster.

$$\Delta VPP(CL_s, CL_t) = \frac{\Delta Vol(CL_s, CL_t)}{k_t} = \frac{Vol(CL_s \oplus CL_t) - Vol(CL_s)}{k_t}, \quad (4.1)$$

$$\Delta EPP(CL_s, CL_t) = \frac{\Delta Edge(CL_s, CL_t)}{k_t} = \frac{Edge(CL_s \oplus CL_t) - Edge(CL_s)}{k_t}. \quad (4.2)$$

We can observe that the functions, ΔVPP and ΔEPP, are not symmetric, that is,

$$\Delta VPP(CL_s, CL_t) \neq \Delta VPP(CL_t, CL_s), \quad \text{and}$$
$$\Delta EPP(CL_s, CL_t) \neq \Delta EPP(CL_t, CL_s).$$

Both functions are used as important criteria to determine the cluster in MDS, into which CL_t is merged.

2.3. Clustering factors

The first clustering algorithm for sequences was proposed in [3], partitioning a sequence into subsequences, each of which is contained in an MBR (or a cluster). This algorithm was slightly modified in [6] to a two-pass algorithm running forward and backward to identify video shot boundaries, and also slightly modified in [7] to support the multidimensional rectangular query. Those algorithms use the marginal cost (MCOST) which is defined as the average number of disk accesses divided by the number of points of the cluster. The grouping of points into clusters is done in such a way that if MCOST increases for the next point of a sequence, then a new cluster is started from the point, otherwise it is included in the current cluster. To determine MCOST, they consider the volume factor based on the volume increment of the current cluster when the next point is included into the cluster. However, it is sometimes not sufficient. We claim the edge of a cluster should also be considered as an important factor in addition to the volume. The following example justifies it intuitively.

Example 1. Let CL_1 and CL_2 be hexahedral clusters with sides a, a, and b $(a < b)$, respectively in the 3-dimensional space as shown in Fig. 1. We are going to determine the cluster into which a point P is merged. In Fig. 1(a), $\Delta VPP(CL_1, CL(P)) = \Delta VPP(CL_2, CL(P)) = a^2 \cdot b$, $\Delta EPP(CL_1, CL(P)) = 4 \cdot a$ and $\Delta EPP(CL_2, CL(P)) = 4 \cdot b$. From the point of volume as the clustering factor, both CL_1 and CL_2 can be candidates. On the other hand, in Fig. 1(b), $\Delta EPP(CL_1, CL(P)) = \Delta EPP(CL_2, CL(P)) = 4 \cdot a$, $\Delta VPP(CL_1, CL(P)) = a^2 \cdot b$, and $\Delta VPP(CL_2, CL(P)) = a^3$. In this case, both CL_1 and CL_2 can also be candidates if we consider the edge as a clustering factor. However, we observe intuitively that CL_1 is appropriate for the former case while CL_2 is good for the latter

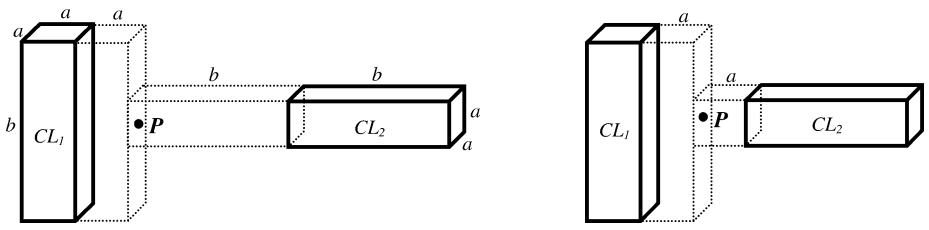

Fig. 1. Clustering factors: volume and edge. (a) Same volumes, different edges. (b) Different volumes, same edges.

case, since $a < b$. It means that both volume and edge are important factors for clustering sequences.

2.4. Measurement of clustering quality

According to the clustering requirements mentioned in Section 1, we use three parameters, *VPP*, *EPP*, and *PPC*, as quantitative measures to evaluate the quality of clustering. Suppose that the clustering process of MDS S with k points generates p clusters. Then, *VPP*, *EPP*, and *PPC* of MDS S are defined as follows:

$$\begin{aligned} VPP &= \frac{\sum_{1\leqslant j\leqslant p} Vol(CL_j)}{\sum_{1\leqslant j\leqslant p} k_j}, \\ EPP &= \frac{\sum_{1\leqslant j\leqslant p} Edge(CL_j)}{\sum_{1\leqslant j\leqslant p} k_j}, \\ PPC &= \frac{\sum_{1\leqslant j\leqslant p} k_j}{p}. \end{aligned} \tag{5}$$

3. Clustering algorithm

Given a collection of MDSs and *minPts*, our algorithm, **CDMDS** (Clustering a Data set of MDS), is to find a set of clusters C and a set of outliers O that optimize the measures of clustering quality defined in Eq. (5). To merge two clusters, our algorithm uses certain criteria that should be satisfied. In this section, we discuss those criteria and our proposed algorithm.

3.1. τ-bounding condition

To determine whether a point cluster is allowed to be merged with another cluster or not, we introduce the bounding condition with respect to the volume and edge of clusters. Prior to discussing it in detail, we first define the concept of the *unit hyper-cube*.

Definition 2 (*Unit hyper-cube*). Let CL_{mds} be a cluster that tightly bounds *all* k points in MDS S. Then, the unit hyper-cube *uCUBE* is defined as the cube, in the space $[0,1]^n$ occupied by a single point when all points are uniformly distributed over the hyper-space of CL_{mds}. If its side-length is e, its volume and edge will be:

$$\begin{aligned} &Vol(uCUBE) = e^n = \frac{Vol(CL_{mds})}{k}, \\ &Edge(uCUBE) \\ &\quad = 2^{n-1}\cdot n\cdot e = 2^{n-1}\cdot n\cdot \sqrt[n]{\frac{Vol(CL_{mds})}{k}}. \end{aligned} \tag{6}$$

If all k points of S are uniformly scattered into the space of cluster CL_{mds}, we can think one point is allocated to a unit hyper-cube. We can figure out intuitively that each point of S forms a cluster whose shape is a unit hyper-cube. However, the uniform distribution is not likely to occur in reality. Points in MDS usually show the clustered distribution in real world. For instance, video frames in a shot are very similar, and thus the points of the same shot are clustered together. The uniform distribution provides the bound in determining whether to merge two clusters or not. The bounding thresholds for volume and edge, τ_{vol} and τ_{edge} for MDS S, are represented as follows:

$$\begin{aligned} \tau_{vol} &= Vol(uCUBE) = e^n, \\ \tau_{edge} &= Edge(uCUBE) = 2^{n-1}\cdot n\cdot e. \end{aligned} \tag{7}$$

Definition 3 (τ-*bounding condition*). Suppose a point cluster CL_t is to be merged with a cluster CL_s in the n-dimensional space. Then, the τ-*bounding condition* is the condition that must be satisfied to merge two clusters and it is defined as follows:

$$\Delta VPP(CL_s, CL_t) \leqslant \tau_{vol} \wedge \Delta EPP(CL_s, CL_t) \leqslant \tau_{edge}. \tag{8}$$

Lemma 1. *The clustering that satisfies the τ-bounding condition guarantees better clustering quality than the case of the uniform distribution, with respect to VPP and EPP.*

Proof. Suppose that after clustering MDS S with k points, p clusters ($0 \leqslant p \leqslant k$) are produced, and the set of clusters is $C = \{CL_1, \ldots, CL_p\}$. Let a cluster CL_j ($1 \leqslant j \leqslant p$) be generated by merging two clusters u times starting with a single point, and the increments of volume, edge and #points by the lth merging be $\Delta Vol_{j,l}$, $\Delta Edge_{j,l}$ and $k_{j,l}$, respectively. Since $k_j = 1$, $Vol(CL_j) = 0$, and $Edge(CL_j) = 0$ in the initial state, after merging u-times, we get:

$$k_j = 1 + \sum_{1 \leqslant l \leqslant u} k_{j,l},$$

$$Vol(CL_j) = \sum_{1 \leqslant l \leqslant u} \Delta Vol_{j,l},$$

and

$$Edge(CL_j) = \sum_{1 \leqslant l \leqslant u} \Delta Edge_{j,l}.$$

Since every merging step satisfies Eq. (8), the following holds: $\Delta VPP_{j,l} \leqslant e^n$ and $\Delta EPP_{j,l} \leqslant 2^{n-1} \cdot n \cdot e$, that is, $\Delta Vol_{j,l} \leqslant k_{j,l} \cdot e^n$ and $\Delta Edge_{j,l} \leqslant k_{j,l} \cdot 2^{n-1} \cdot n \cdot e$. Thus, we derive:

$$\begin{aligned} Vol(CL_j) &= \sum_{1 \leqslant l \leqslant u} \Delta Vol_{j,l} \leqslant \sum_{1 \leqslant l \leqslant u} (k_{j,l} \cdot e^n) \\ &< \left(1 + \sum_{1 \leqslant l \leqslant u} k_{j,l}\right) \cdot e^n = k_j \cdot e^n, \end{aligned} \tag{9.1}$$

$$\begin{aligned} Edge(CL_j) &= \sum_{1 \leqslant l \leqslant u} \Delta Edge_{j,l} \\ &\leqslant \sum_{1 \leqslant l \leqslant u} (k_{j,l} \cdot 2^{n-1} \cdot n \cdot e) \\ &< \left(1 + \sum_{1 \leqslant l \leqslant u} k_{j,l}\right) \cdot 2^{n-1} \cdot n \cdot e \\ &= k_j \cdot 2^{n-1} \cdot n \cdot e. \end{aligned} \tag{9.2}$$

Suppose that the volume and the edge per point of MDS S after clustering are VPP and EPP, respectively, and those of the uniform distribution are VPP_0 and EPP_0, respectively. Using Eqs. (5), (9.1), (9.2), we conclude:

$$\begin{aligned} VPP &= \frac{\sum_{1 \leqslant j \leqslant p} Vol(CL_j)}{\sum_{1 \leqslant j \leqslant p} k_j} \\ &= \frac{Vol(CL_1) + \cdots + Vol(CL_p)}{k_1 + \cdots + k_p} \\ &< \frac{k_1 \cdot e^n + \cdots + k_p \cdot e^n}{k_1 + \cdots + k_p} = e^n = VPP_0, \end{aligned} \tag{10.1}$$

$$\begin{aligned} EPP &= \frac{\sum_{1 \leqslant j \leqslant p} Edge(CL_j)}{\sum_{1 \leqslant j \leqslant p} k_j} \\ &= \frac{Edge(CL_1) + \cdots + Edge(CL_p)}{k_1 + \cdots + k_p} \\ &< \frac{k_1 \cdot 2^{n-1} \cdot n \cdot e + \cdots + k_p \cdot 2^{n-1} \cdot n \cdot e}{k_1 + \cdots + k_p} \\ &= 2^{n-1} \cdot n \cdot e = EPP_0. \qquad \square \end{aligned} \tag{10.2}$$

3.2. θ-merging condition

After clusters are initially generated from a sequence of points, those clusters that are spatially close need to be merged together. To determine whether two clusters are allowed to be merged or not, we introduce the θ-*merging condition* as follows:

Definition 4 (θ-*merging condition*). Suppose two clusters CL_s and CL_t are to be merged together in the n-dimensional space. Let $\theta_{vol} = Vol(CL_s) + Vol(CL_t)$ and $\theta_{edge} = Edge(CL_s) + Edge(CL_t)$, then the θ-*merging condition* is the condition that must be satisfied to merge two clusters and it is defined as follows:

$$Vol(CL_s \oplus CL_t) \leqslant \theta_{vol} \wedge Edge(CL_s \oplus CL_t) \leqslant \theta_{edge}. \tag{11}$$

Lemma 2. *The merging of two clusters, CL_s and CL_t, that satisfies the θ-merging condition guarantees better clustering quality than*
(1) *the case of the uniform distribution, with respect to VPP and EPP, and*
(2) *the case of non-merging with respect to VPP, EPP, and PPC.*

Proof. Let $CL_m = CL_s \oplus CL_t$, then $k_m = k_s + k_t$. Let VPP_m, VPP_s, VPP_t be the VPPs of CL_m, CL_s, CL_t, respectively, and EPP_m, EPP_s, EPP_t be the EPPs of CL_m, CL_s, CL_t, respectively. By Eqs. (2), (11), we obtain

$$k_m \cdot VPP_m \leqslant k_s \cdot VPP_s + k_t \cdot VPP_t \quad \text{and}$$
$$k_m \cdot EPP_m \leqslant k_s \cdot EPP_s + k_t \cdot EPP_t.$$

Case (1). Since $VPP_s < e^n$, $VPP_t < e^n$, $EPP_s < 2^{n-1} \cdot n \cdot e$, and $EPP_t < 2^{n-1} \cdot n \cdot e$ hold by Eqs. (9.1), (9.2), we get:

$$\begin{aligned} VPP_m &\leqslant \frac{k_s \cdot VPP_s + k_t \cdot VPP_t}{k_m} \\ &< \frac{k_s \cdot e^n + k_t \cdot e^n}{k_m} = e^n, \end{aligned}$$

$$\begin{aligned} EPP_m &\leqslant \frac{k_s \cdot EPP_s + k_t \cdot EPP_t}{k_m} \\ &< \frac{k_s \cdot 2^{n-1} \cdot n \cdot e + k_t \cdot 2^{n-1} \cdot n \cdot e}{k_m} \\ &= 2^{n-1} \cdot n \cdot e. \end{aligned}$$

Case (2). Let VPP_n and EPP_n be the VPP and EPP of CL_s and CL_t for the case of non-merging. Then, by Eqs. (5), (11), the following holds:

$$\begin{aligned} VPP_m &= \frac{Vol(CL_m)}{k_m} = \frac{Vol(CL_s \oplus CL_t)}{k_s + k_t} \\ &\leqslant \frac{Vol(CL_s) + Vol(CL_t)}{k_s + k_t} = VPP_n. \end{aligned}$$

In a similar way, $EPP_m \leqslant EPP_n$. It is clear that merging two clusters has better quality with respect to *PPC* since *PPC* without merging is $(k_s + k_t)/2$ while that with merging is k_m. Therefore, we conclude Lemma 2 holds. □

3.3. Determining outliers

A sequence may contain outliers that are considered to be unimportant with respect to the overall clustering pattern. In our method, each point in a sequence is initially regarded as a cluster with a single point, and then closely related clusters are repeatedly merged. If a certain cluster has "*far fewer*" points than the average after the clustering process, all points in it can then be thought as outliers. For instance, if a cluster with 2 or 3 points is located away from other clusters, it may be a set of outliers with a high possibility. "*Far fewer*" is of course heuristically determined depending on applications. Too small value of *minPts* makes unimportant clusters be indexed, degrading the memory utilization, while too large value of *minPts* makes meaningful clusters be missed. In this context, if a cluster has points the number of which is less than a given *minPts* value after the clustering process, all points in the cluster are regarded as outliers. Those outliers are not indexed, but written out to disk for later processing.

3.4. Algorithm description

Algorithm **CDMDS** in Fig. 2 describes the overall structure. Algorithm **ChooseCluster** determines the cluster into which a point is to be merged. First, each cluster is evaluated with respect to τ-*bounding condition* to determine candidate clusters. Then, it chooses the most appropriate one among the candidates. To do it, we introduce the combined measure λ_{join} considering weight factors, w_1 and w_2, as follows:

$$\lambda_{join}\big(CL_s, CL(P_t)\big) = \frac{w_1 \cdot \lambda_{vol} + w_2 \cdot \lambda_{edge}}{w_1 + w_2},$$

where

$$\lambda_{vol} = \frac{\Delta VPP(CL_s, CL(P_t))}{Vol(CL_s \oplus CL(P_t))} \quad \text{and}$$
$$\lambda_{edge} = \frac{\Delta EPP(CL_s, CL(P_t))}{Edge(CL_s \oplus CL(P_t))}. \tag{12}$$

We can choose values of w_1 and w_2 depending on the importance of the volume and edge factor. In our experiment, we use the same importance, that is, $w_1 = w_2 = 1$. Normalized values, λ_{vol} and λ_{edge}, are introduced to compensate for the variance between ΔVPP and ΔEPP since the simple summation of two values with a big difference usually leads to unexpected results that a small value is nearly ignored. Among candidate clusters, the cluster that has the minimum λ_{join} is selected for subsequent merging. If

Algorithm CDMDS
***Input*:** a data set of *MDSs*, *minPts*
***Output*:** cluster set C, outlier set O
Step 0: set $C \leftarrow \phi$, set $O \leftarrow \phi$
Step 1: **for each** *MDS* S_i in the data set $(1 \leqslant i \leqslant N)$
C_i, $O_i \leftarrow$ **ClusterMDS**(MDS S_i, *minPts*)
$C \leftarrow C \cup C_i$, $O \leftarrow O \cup O_i$
end for
Step 2: **return** set C, set O

Algorithm ChooseCluster
***Input*:** point P_m, set C_i, τ_{vol}, and τ_{edge}
***Output*:** ϕ or cluster CL_c
Step 0:
$result \leftarrow \phi$, $\lambda_{current} \leftarrow \infty$, $vol_{current} \leftarrow \infty$
Step 1: /* Choose a cluster to merge with */
if set $C_i == \phi$ **then**
return *result*
for each cluster CL_c in set C_i
compute $\Delta VPP(CL_c, CL(P_m))$,
$\Delta EPP(CL_c, CL(P_m))$
if τ*-bounding condition holds* **then**
compute $\lambda_{join}(CL_c, CL(P_m))$
if $(\lambda_{join}(CL_c, CL(P_m)) < \lambda_{current}) \vee$
$(\lambda_{join}(CL_c, CL(P_m)) == \lambda_{current} \wedge$
$Vol(CL_c \oplus CL(P_m)) < vol_{current})$ **then**
$\lambda_{current} \leftarrow \lambda_{join}(CL_c, CL(P_m))$
$vol_{current} \leftarrow Vol(CL_c \oplus CL(P_m))$
$result \leftarrow CL_c$
end if
end if
end for
Step 2: **return** *result*

Algorithm ClusterMDS
***Input*:** MDS S_i, *minPts*
***Output*:** set C_i, set O_i
Step 0:
set $M \leftarrow$ all points in S_i
set $C_i \leftarrow \phi$, set $O_i \leftarrow \phi$, $numCL \leftarrow 0$
compute τ_{vol} and τ_{edge} for S_i
Step 1: /* Initial cluster generation */
for each point P_m in set M
$CL_c \leftarrow$ **ChooseCluster**$(P_m, C_i, \tau_{vol}, \tau_{edge})$
if $CL_c == \phi$ **then**
represent P_m in the cluster form CL_{numCL}
$C_i \leftarrow C_i \cup \{CL_{numCL}\}$
$numCL \leftarrow numCL + 1$
else
$CL_c \leftarrow CL_c \oplus CL(P_m)$
end if
end for
Step 2: /* Cluster elaboration */
for each pair of cluster (CL_x, CL_y) in C_i
if θ*-merging condition holds* **then**
$CL_x \leftarrow CL_x \oplus CL_y$
remove CL_y from set C_i
end if
end for
Step 3: */ Outlier determination */
for each cluster CL_c in C_i
if $k_c <$ *minPts* **then**
remove CL_c from C_i
insert all points of CL_c into O_i
end if
end for
Step 4: **return** set C_i, set O_i

Fig. 2. Clustering algorithm.

there exist multiple clusters with the same λ_{join}, then the cluster CL_s with the smallest value of $Vol(CL_s \oplus CL(P_t))$ is selected. Algorithm **ClusterMDS** performs the clustering of each MDS. In Step 1, a set of clusters is initially generated from a sequence of points by using Algorithm **ChooseCluster**, without considering the order of clusters in a sequence. In Step 2, every pair of clusters is evaluated with respect to θ*-merging condition* to determine whether each pair of clusters is to be merged or not. A pair that satisfies the condition is merged together. In Step 3, the number of points in each cluster is tested if it is less than *minPts* as described in Section 3.3. All outliers identified are moved to the set of outliers.

3.5. Complexity of the algorithm

By Algorithm **CDMDS** in Fig. 2, it is clear that its complexity is $O(N)$ with respect to the number of sequences N in a database. As for the numbers of points and clusters in each sequence, m and c respectively, we observe in Algorithm **ClusterMDS** that the complexity is the order of mc in Step 1, the order of c^2 in Step 2, and the order of c in Step 3. Since c is negligibly small compared to m and the value of c

varies inside the algorithm, we ignore c. Consequently, the complexity of the algorithm is $O(Nm)$, which is linear with respect to N and m.

4. Experiments and concluding remarks

4.1. Test data sets

In order to evaluate the effectiveness of our algorithm, we have conducted experiments on video data sets such as TV news, dramas, and animation films. Video streams of different lengths are generated from the video data sources. We extracted RGB color features from each frame of the video streams, and represented a frame by a point in the $[0, 1]^3$ space according to its RGB values. Thus, each video stream is represented by an MDS. Table 1 summarizes test data sets used in the experiment.

4.2. Results

Our experiment focuses on showing the clustering quality of the algorithm with respect to VPP, EPP, and PPC. We compared our algorithm to the MCOST algorithm proposed in [3] since other related algorithms are based on it with slight modifications. Our algorithm produces clusters, not considering their order in an MDS, while the MCOST algorithm considers it, because the goal of our algorithm is to represent an MDS by as few clusters as possible. To determine outliers, a parameter *minPts* is set to 5. We believe this choice is reasonable in the video application domain, since it takes approximately 1/6 s to play a 5-frame video and the clusters with frames less than 5 are nearly meaningless in reality. A 1/6 s playing time may not be perceived well by the human visual system. The experimental results are shown in Fig. 3 and Table 2.

Fig. 3 shows that VPPs of both MCOST and CDMDS decrease as the length of video sequences increases. But VPP of CDMDS is 11 to 23% of that of MCOST. In Fig. 4, we can observe that EPP of CDMDS is also 38 to 59% of that of MCOST. It means the clusters produced by CDMDS algorithm are denser in their volume and edge, which makes the subsequent search more efficient. As another indicator to show the effectiveness of clustering, we have listed PPC's of CDMDS and MCOST in Table 2. The ratio CDMDS/MCOST is 0.69 to 2.67 and increases as the length of sequences increases. It shows that the clusters generated by our algorithm become gradually denser for larger video streams, since a longer video

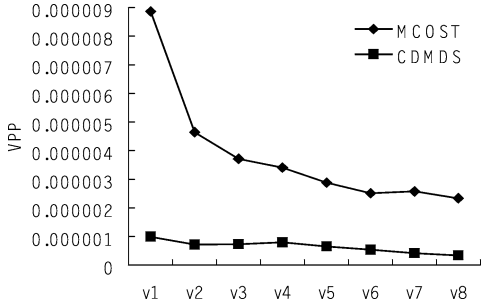

Fig. 3. VPP comparison.

Table 1
Test data sets used in the experiment

Data set name	Sequence length	Number of sequences
v1	$L \leqslant 64$	1632
v2	$64 < L \leqslant 128$	1401
v3	$128 < L \leqslant 256$	1206
v4	$256 < L \leqslant 512$	1017
v5	$512 < L \leqslant 1024$	912
v6	$1024 < L \leqslant 2048$	813
v7	$2048 < L \leqslant 4096$	651
v8	$4096 < L$	513
Total		8145

Table 2
PPC ratio (CDMDS/MCOST)

Video type	PPC (MCOST)	PPC (CDMDS)	CDMDS/MCOST ratio
v1	62.73	43.17	0.6882
v2	74.74	64.60	0.8643
v3	83.34	91.53	1.0982
v4	85.24	119.77	1.4051
v5	85.92	150.48	1.7515
v6	85.23	172.34	2.0221
v7	83.80	193.04	2.3037
v8	86.10	230.30	2.6748

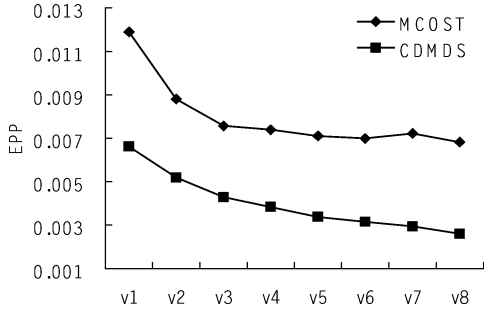

Fig. 4. EPP comparison.

Table 3
Outlier ratio of MDS

Video type	Outlier ratio (#outlier / #point)
v1	0.06278
v2	0.03812
v3	0.02541
v4	0.02108
v5	0.01651
v6	0.01585
v7	0.01549
v8	0.01294

clip has higher probability that it has similar shots. For instance, a news video clip has many similar shots that the same anchor appears. Table 3 indicates the ratio of the number of outliers vs. the number of points in sequences. It decreases from 6.28 to 1.29% as the sequences become longer. Those outliers are treated separately from clusters for similarity search processing.

4.3. Concluding remarks

In this paper, we have investigated clustering of MDS such as video streams. To solve the problem, we have proposed an effective algorithm which meets five special requirements identified in Section 1. Our algorithm shows considerable effectiveness with respect to VPP, EPP, and PPC as shown in the experiment. In addition, it deals with outliers appropriately, and requires only one input parameter *minPts*, except an MDS data set itself. One of potential applications which is emphasized in this paper is the clustering of video data sets, but we believe other application areas can also benefit.

Acknowledgement

We would like to thank Professor Francis Y.L. Chin, editor, for his help and reviewers for their valuable comments. This research was supported by Korea Research Foundation Grant (KRF-2000-041-E00262).

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: Proc. of ACM SIGMOD Internat. Conference on Management of Data, 1998, pp. 94–105.
[2] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proc. Internat. Conference on Knowledge Discovery in Databases and Data Mining, 1996, pp. 226–231.
[3] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, Fast subsequence matching in time-series databases, in: Proc. of ACM SIGMOD Internat. Conference on Management of Data, 1994, pp. 419–429.
[4] S. Guha, R. Rastogi, K. Shim, CURE: An efficient clustering algorithm for large databases, in: Proc. of ACM SIGMOD Internat. Conference on Management of Data, 1998, pp. 73–84.
[5] A. Guttman, R-trees: A dynamic index structure for spatial searching, in: Proc. of ACM SIGMOD Internat. Conference on Management of Data, 1984, pp. 47–57.
[6] V. Kobla, D. Doermann, C. Faloutsos, Video Trails: Representing and visualizing structure in video sequences, in: Proc. of ACM Multimedia, 1997, pp. 335–346.
[7] S.L. Lee, S.J. Chun, D.H. Kim, J.H. Lee, C.W. Chung, Similarity search for multidimensional data sequences, in: Proc. of IEEE Internat. Conference on Data Engineering, 2000, pp. 599–608.
[8] R.T. Ng, J. Han, Efficient and effective clustering methods for spatial data mining, in: Proc. of the VLDB Conference, 1994, pp. 144–155.
[9] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: An efficient data clustering method for very large databases, in: Proc. of ACM SIGMOD Internat. Conference on Management of Data, 1996, pp. 103–114.