MCF tree-based Clustering Method for Very Large Mixed-type Data Set

HYEONG-CHEOL RYU¹ AND SUNGWON JUNG¹

¹Department of Computer Science and Engineering, Sogang University, 35 Baekbeom-ro, Mapo-gu, Seoul 121-742, Republic of Korea (e-mail: hchryu@gmail.com and jungsung@sogang.ac.kr)

Corresponding author: Sungwon Jung (e-mail: jungsung@sogang.ac.kr).

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2015R1A1A01058001) and the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2017-0-01628) supervised by the IITP (Institute for Information & communications Technology Promotion).

ABSTRACT Several clustering methods have been proposed for analyzing numerous mixed-type data sets composed of numeric and categorical attributes. However, existing clustering methods are not suitable for clustering very large mixed-type data sets because they require a high computational cost or a large memory size. We propose a novel clustering method for very large data sets using a mixed-type clustering feature (MCF) vector with summary information about a cluster. The MCF vector consists of the CF vector and a histogram to summarize the mixed-type values. Based on the MCF vector, we propose an MCF tree, along with a distance measure between the MCF vectors representing two clusters. Unlike previous studies that summarize a data set based on a fixed memory size, we estimate a small initial memory size for building the tree. Then, the memory size is adaptively increased to estimate a more accurate threshold by reflecting the size reduction in the re-built tree. Our theoretical analysis demonstrates the efficiency of the proposed approach. Experimental results on very large synthetic and real data sets demonstrate that the proposed approach clusters the data sets significantly faster than existing clustering methods while maintaining similar or better clustering accuracy.

INDEX TERMS Clustering, CF² tree, MCF tree, Numeric data, Categorical data, Mixed-type data, Very Large Data sets.

I. INTRODUCTION

CLUSTERING groups a data set into clusters based on the concepts of distance or similarity. It is widely used in a diverse range of fields (e.g., data science, machine learning, pattern recognition, and artificial intelligence) as an effective technique for data analysis. The attributes of real-world data sets are commonly composed of both numeric and categorical attributes, e.g., health, marketing, and medical. Numeric attributes can take real values, e.g., age and height. Categorical attributes can take the values in a fixed number of categories, e.g., sex and religion. A “mixed-type” data set is composed of both numeric and categorical attributes.

There are two types of commonly used clustering methods: partitional clustering and hierarchical clustering. Partitional clustering methods partition a data set into final clusters, each of which is represented by its cluster center. In this paper, the term “final cluster” indicates “the cluster found by the clustering methods,” whereas the term “true cluster” indicates “the original cluster with the ground-truth label.” Hierarchical clustering methods generate a hierarchy of clusters in either a top-down or bottom-up sequence.

Both types of clustering methods for mixed-type data sets have similar disadvantages to those for non-mixed-type data sets (e.g., numeric data sets). Partitional clustering methods for mixed-type data sets cannot achieve very large data sets rapidly because they repeat the assignment and update steps by using whole data sets until their stopping criteria are satisfied. Specifically, if all data do not reside in the main memory, disk I/Os are required repeatedly in each iteration. This drastically increases the clustering time. Many hierarchical methods for mixed-type data sets have been proposed. However, these hierarchical clustering methods require a proximity matrix, and generating the matrix requires high computational costs. Further, a large amount of memory must be allocated to maintain the matrix until the clustering is completed. This means that these hierarchical clustering...
methods are not suitable for clustering very large mixed-type data sets.

Sampling-based [36], dimensionality reduction-based, and summary-based clustering methods [13] are broadly used for clustering large data sets [35]. However, deriving appropriate samples from data sets for sampling-based clustering methods is a challenge [30]. Further, dimensionality reduction-based methods can lose the information that maintains the significant properties of the data sets [8], [14]. Therefore, we focus on summary-based clustering methods for the analysis of large data sets.

Summary-based clustering methods for numeric data sets have been proposed to cluster very large data sets effectively. BIRCH (balanced iterative reducing and clustering using hierarchies) and CF+ -ERC (clustering features+ tree-based effective multiple range queries-based clustering) are examples of this type of clustering method [35], [38], [39]. However, these clustering methods cannot cluster mixed-type data sets because they only handle numeric data sets.

A summary-based clustering method for mixed-type data sets, which is an extension of the CF tree of BIRCH, was proposed in [13]. They computed a distance between data using their distance measure based on the log-likelihood function. If the distance between data was less than a threshold, the data were summarized in the CF vector. However, their threshold estimation method is limited by a given memory size, similar to BIRCH. If a given memory size is not suitable for clustering mixed-type data sets, their threshold estimation method does not perform well, thereby giving poor final clusters.

In this paper, we propose a novel summary-based clustering method for very large mixed-type data sets by incorporating a dynamically increasing memory size for more accurate threshold estimation. For this, we first introduce the mixed-type clustering features (MCF) vector that the numeric and categorical values of the mixed-type objects in the cluster are summarized using the CF vector [35] and the histogram proposed in K-histograms [19], respectively. Next, we implement a mixed-type clustering feature (MCF) tree, which is an extension of the CF+ tree [35].

We generalize the distance measure of K-histograms to compute the distance between histograms of the categorical values of two clusters. Therefore, our proposed distance measure only uses the MCF vector to compute the precise distance between two clusters. The MCF vector is also used to obtain a diameter of its corresponding cluster. Unlike previous works [13], [35], [38], [39], our method first estimates a small initial memory size for building the MCF tree by considering the size and characteristics of a given mixed-type data set. It then adaptively increases the memory size for the MCF tree to estimate a more accurate threshold. The memory size is adaptively increased by considering the size reduction of the re-built MCF tree so that the threshold is not increased too conservatively or too aggressively.

The MCF tree provides a set of microclusters that have summary information about the overall data set. Subsequently, existing mixed-type clustering methods (e.g., convex k-means and W-K-prototypes) are used in our proposed approach as the global clustering methods to group a set of microclusters into the final clusters. Consequently, our summary-based clustering method clusters mixed-type data sets more rapidly than existing clustering methods without sacrificing accuracy compared to non-summary-based clustering methods. Furthermore, the range-query-based clustering method ERC (effective multiple range queries-based clustering), which uses the structure of the tree, can also be applied to the MCF tree. Therefore, various existing clustering methods can be applied to our global clustering method.

The main contributions of our proposed method are summarized as follows:

- We propose a scheme for summarizing mixed-type data sets for clustering using an MCF vector composed of the CF vector and histogram.
- We develop an efficient distance computation method between MCF vectors.
- We develop an adaptive memory size increasing method to estimate more accurate thresholds for constructing MCF trees.
- We propose a method of applying the range-query-based clustering method ERC to MCF Tree as a global clustering method.

We use Figure 1 to explain a process of our clustering method briefly. First, we build an MCF tree as follows. We set a threshold $T = 0$ and an initial memory size is estimated.
An MCF tree $Tr$ with $T$ is created. For each object $o$, $o$ is inserted into $Tr$ if $o$ is not in $Tr$. If the size of $Tr$ is greater than the current given memory size, we increase the threshold $T$. With these increased threshold values, we rebuild a new MCF tree $Tr'$ from the current MCF tree $Tr$ by inserting all leaf entries into $Tr'$. At that time, each entry of $Tr'$ can absorb more objects because of an increase in $T$. After the completion of rebuilding the MCF tree, the size of $Tr'$ becomes smaller than that of $Tr$. We increase the memory size by using our proposed scheme. We now make $Tr'$ as our current MCF tree by resetting $Tr = Tr'$. Our current MCF tree $Tr$ is ready to get the insertion of the remaining objects from the data set. All above steps are repeated until all the objects are inserted into the MCF tree. After building the MCF tree, existing mixed-type clustering methods group a set of microclusters into the final clusters. The detailed process of our method will be discussed in Section IV.

The remainder of this paper is organized as follows. Section II handles the CF$^+$-ERC and clustering methods for mixed-type data sets in detail. In Section III, an MCF vector consisting of summary information about a cluster is described. We explain that the MCF tree identifies microclusters and how to apply ERC to our method in Section IV. Theoretical and extensive experimental analyses of our method are presented in Section V. Finally, Section VI provides our concluding remarks.

II. RELATED WORKS

In this section, we briefly explain CF$^+$-ERC [35], which is an extension of BIRCH [38], and existing clustering methods for mixed-type data.

A. CF$^+$ TREE

We share the definitions of the CF vector, distance measure, and CF$^+$ tree [35].

Definition II.1. Given $N$ $d$-dimensional objects in a cluster $X$: $\{\alpha_i^X\}$, where $i = 1, 2, \ldots, N$, the CF vector of $X$ is defined as $CF = (N, LS, SS)$, where $N$ is the number of objects in $X$; $LS$ is the linear sum of the $N$ objects, that is, $LS = \sum_{i=1}^{N} \alpha_i^X$; and $SS$ is the sum of the $N$ squared objects, that is, $SS = \sum_{i=1}^{N} \alpha_i^{X2}$.

Definition II.2. Given $N$ $d$-dimensional objects in a cluster $X$: $\{\alpha_i^X\}$, where $i = 1, 2, \ldots, N$, the centroid $c$ and the average radius $ar$ are obtained using the following formula:

\[
\begin{align*}
c(X) &= \frac{\sum_{i=1}^{N} \alpha_i^X}{N} = \frac{LS}{N} \\
ar(X) &= \sqrt{\frac{\sum_{i=1}^{N} (\alpha_i^X - c(X))^2}{N}} \\
&= \sqrt{\frac{N \cdot c(X)^2 + SS - 2 \cdot c(X) \cdot LS}{N}}.
\end{align*}
\]

Definition II.3. Given two clusters $X$ and $Y$ having CF vectors $CF_X = (N_X, L\bar{S}_X, SS_X)$ and $CF_Y = (N_Y, L\bar{S}_Y, SS_Y)$, respectively, the centroid Euclidean distance $d_0$ between $X$ and $Y$ is obtained by the following formula:

\[
d_0(X, Y) = |c(X) - c(Y)|.
\]

Definition II.4. The CF$^+$ tree satisfies the followings:

1) Each non-leaf node contains, at most, $C$ entries in the form $[CF_i, child_i]$, where $i = 1, 2, \ldots, C$; $child_i$ is a pointer to its $i$-th child node, and $CF_i$ is the CF vector of the subcluster pointed to by $child_i$. An entry of the non-leaf node is called a subcluster.

2) Each leaf node contains, at most, $L$ entries in the form $[CF_i]$, where $i = 1, 2, \ldots, L$. An entry of the leaf node is called a microcluster.

3) All leaf nodes are chained together for efficient sequential scanning using the “prev” and the “next” pointers.

CF$^+$-ERC obtains microclusters by building a CF$^+$ tree using a threshold. The CF$^+$ tree is built as follows: For each object $o$, starting from the root node, it recursively goes down to the closest child node on the CF$^+$ tree using the distance measures such as $d_0$. For example, given a cluster $X$ and an object $o$, $d_0(X, o)$ is the centroid Euclidean distance between the centroids of $X$ and $o$. When $o$ arrives at a leaf node $nl$, if the distance between $o$ and its closest entry (i.e., microcluster $mc$) of $nl$ is less than a given threshold, $o$ is absorbed in $mc$. Otherwise, $o$ generates a new entry, and the new entry is inserted into $nl$. If there is space in $nl$ for this new entry, the new entry is inserted into $nl$. Otherwise, $nl$ must be split. Node splitting is performed by the adaptable node split scheme [35]. In the adaptable node split scheme, all entries of $nl$ are divided into two entry sets as follows. First, the farthest entry pair of $nl$ is chosen as the seeds. Next, each seed is moved to each entry set. After that, each of the remaining entries is distributed to the closest entry set. At that time, the remaining entries use the distances from the CF vectors of the entry sets rather than those of the seeds. This is because the CF vector of each entry set reflects all entries of the entry set.

Splitting $nl$ causes the insertion of a new non-leaf entry into the parent node, and then the parent node can also be split. Thus, the node split can be propagated to the parent node, and it can propagate up to the root node. If a non-leaf node $nl_{ln}$ has room for the new entry created from the propagation of the node split, the propagation of the node split stops at $nl_{ln}$. A split of the root node increases the height of the CF$^+$ tree by one.

If the size of the CF$^+$ tree is less than that of a given memory size, CF$^+$-ERC rebuilds the tree using the average value of the closest distances of all leaf nodes as the new threshold. Tree rebuilding continues until the size of the entire CF$^+$ tree is less than that of a given memory size. Note that the CF$^+$ tree aims to summarize the data set so that the size of the summary is less than the memory size.
Definition II.5. Given two microclusters $MC_x$ and $MC_y$ having their centroids $c(MC_x)$ and $c(MC_y)$, and their radii $ar(MC_x)$ and $ar(MC_y)$, respectively, inter-microcluster distance between $MC_x$ and $MC_y$ is defined as

$$IMD(MC_x, MC_y) = d_0(c(MC_x), c(MC_y)) - (ar(MC_x) + ar(MC_y)).$$

If the IMD between two microclusters is less than $T$, both microclusters reside in the same final cluster. ERC consists of two steps: partition and refinement. The partition step groups the linearly adjacent microclusters within $T$ into a segment called a microcluster segment ($MCS$) using the microcluster linearization property of the CF+ tree.

For instance, in Figure 2, the CF+ tree has 13 microclusters ($MC_1, MC_2, \ldots, MC_{13}$) that are linearly adjacent. Because $IMD(MC_1, MC_2)$ ($IMD$ between $MC_3$ and $MC_4$) is less than $T$, $MC_3$, and $MC_4$ are grouped into $MCS_3$. Similarly, six $MCS$s, $MCS_1, MCS_2, \ldots, MCS_6$ are found in the partition step.

In the refinement step, the effective refinement step ($ERS$) of ERC gathers a set of $MCS$s into the set of final clusters using multiple range queries. For each $MCS$, $ERS$ calls an effective range query ($ERQ$), which uses the structure of the CF+ tree to discover the connections between $MCS$ and others without redundant query processing. A connection between $MCS_i$ and $MCS_j$ means that at least one $IMD$ exists between $MC_x$ and $MC_y$ is less than $T$, where $MC_x \in MCS_i$ and $MC_y \in MCS_j$. Thus, $MCS_i$ and $MCS_j$ are merged into the same final cluster. As a result, ERC returns a set of final clusters.

In Figure 2, $IMD(MC_2, MC_4)$, and $IMD(MC_5, MC_8)$ are less than $T$. However, these two microcluster pairs are not linearly adjacent; hence, the partition step does not group them into the two respective $MCS$s. The $ERS$ of

---

**FIGURE 2:** CF+ tree constructed using 60 objects

- **node_1:** SC, SC_2, SC_3
- **node_2:** MC, MC_2, MC_3
- **node_3:** MC, MC_2, MC_4, MC_5
- **node_4:** MC, MC_3, MC_4, MC_5, MC_6

![Image of the CF+ tree constructed using 60 objects](image-url)
TABLE 1: Clustering methods for mixed-type data sets

| Method                     | Characteristics                                      |
|----------------------------|------------------------------------------------------|
| Ahmad [5], Modha [32], Huang [24], Ji [27], Brnaow [10] | Partitional clustering methods. Not suitable for clustering very large data sets because they have to load the entire data sets into their memory to cluster. |
| HajKacem [17], Hakacem [18], Jridi [28] | Distributed clustering methods based on the specific distributed processing environments. |
| Chiu [13] | Summary-based clustering method using a fixed memory size. |
| Our approach | Summary-based clustering method using the adaptive memory-size-increasing scheme. Improve the clustering accuracy and reduce clustering time significantly. |

C. CLUSTERING METHODS FOR MIXED-TYPE DATA SETS

Most of the existing clustering methods cluster mixed-type data sets in partitional ways [5], [10], [24], [27], [32]. However, these methods are not suitable for clustering very large data sets because they have to load the entire data sets into their memory to cluster. Distributed clustering methods based on the specific distributed processing environments, i.e., MapReduce [15] and Spark [37] frameworks, cluster very large data sets rapidly [17], [18], [28]. The existing summary-based clustering method efficiently analyzes very large data sets [13], but its clustering performance is determined by a fixed memory size. Table 1 shows the main characteristics of these methods in comparison with our proposed scheme. In Table 1, the partitional clustering methods for mixed-type data sets [5], [10], [24], [27], [32] are used for the comparative global clustering methods of our proposed MCF-tree. Thus, we give more detailed description of these partitional clustering methods in the following two paragraphs.

A new distance measure to compute the dissimilarity between mixed-type objects were proposed [5]. The authors compute the distance between two attribute values of a categorical attribute from the data sets. The distance depends upon the co-occurrence of these attribute values with the attribute values of other attributes. The weights of numeric attributes are also calculated in this method such that more significant attributes are given greater weights. A convex K-means for the data in heterogeneous feature spaces was proposed [32]. They normalized each numeric attribute and used 1-in-q representation for each q-ary categorical attribute. In order to compute the distortion between a centroid and an object, they applied the squared Euclidean distance for the numeric feature and the cosine distance for the categorical feature. The distortion in each feature is multiplied by the feature weight and summed to the total distortion.

W-K-prototypes was proposed [24]. In each iteration of W-K-prototypes, the attribute weights are updated and used in the cost function. These weights are inversely proportional to the sum of the within-cluster distances. An extension of W-K-prototypes was proposed [27]. Its new cost function reflected the significance of the cluster center and the weight of W-K-prototypes. The significance of an attribute is initially selected randomly, followed by an update to its value for each iteration. The random selection of the significance of an attribute can worsen the problem of random initialization of the cluster center because it would lead to different results in different runs. The K-mixed prototypes further distinguished the categorical types of attributes into nominal versus binary types [10]. The K-mixed prototypes increased its clustering accuracy by assigning more significance to the attributes of the binary types than those of nominal types.

III. MIXED-TYPE CLUSTERING FEATURES AND MEASURES

In this section, we introduce the MCF vector for summarizing clusters. Our proposed distance measure computes the distance between two clusters using the MCF vectors. For computing the distance between two clusters considering only categorical attributes, we generalize the distance measure of K-histogram to the distance measure between histograms of the categorical attributes of two clusters.

A. NOTATIONS

Suppose a mixed-type data set with \( d \) attributes composed of \( d' \) numeric and \( d'' \) categorical attributes (\( d = d' + d'' \)). Let \( A_i = \{a_1, \ldots, a_m\} \) be a set of \( m \) distinct values of the \( l \)-th categorical attribute. It should be noted that \( m \) may vary depending on the domain characteristics of the attribute. Let \( o_i = (a'_i, o''_i) \) denote the \( i \)-th mixed-type object with \( d \) attributes (\( d' \) numeric and \( d'' \) categorical attributes). Here, \( o'_i = (o'_i,1, o'_i,2, \ldots, o'_i,d') \) and \( o''_i = (o''_{i,1}, o''_{i,2}, \ldots, o''_{i,d''}) \), where \( o''_{i,l} \) is the value of the \( l \)-th numeric attribute of \( o''_i \), and \( o''_{i,l} \) is the value of the \( l \)-th categorical attribute of \( o''_i \) (i.e., \( o''_{i,l} \in A_l \)). Let \( X = \{o_1, \ldots, o_N\} \) be a cluster. Then, we can derive two types of sets, \( \mathcal{X}' \) and \( \mathcal{X}'' \), from \( X \), where \( \mathcal{X}' = \{o'_{1,1}, o'_{2,1}, \ldots, o'_{N,1}\} \) and \( \mathcal{X}'' = \{o''_{1,1}, o''_{2,1}, \ldots, o''_{N,1}\} \). Let \( \mathcal{X}'_i = \{o'_{i,1}, o'_{i,2}, \ldots, o'_{i,4}\} \) be a set of values of the \( l \)-th categorical attribute of all the objects in \( \mathcal{X}' \). Let \( V_i \) be a set of distinct values in \( \mathcal{X}'_i \), and \( g(a_j, \mathcal{X}'_i) \) be the frequency of value \( a_j \) in \( \mathcal{X}'_i \). Let \( H_{\mathcal{X}'_i} = \{ \langle a_j, f_j \rangle \mid a_j \in V_i, f_j = g(a_j, \mathcal{X}'_i) \} \). We denote \( H_{\mathcal{X}''} \) briefly by \( H_x \). The histogram set of \( \mathcal{X}'' \) is defined as \( H_x = (H_{\mathcal{X}'_1}, H_{\mathcal{X}'_2}, \ldots, H_{\mathcal{X}'_d}) \). The symbolic notation is summarized in Table 2.
TABLE 2: Symbolic notation

| Symbolic notation | Description |
|------------------|-------------|
| $d = d^e + d^c$ | Number of numeric attributes |
| $d^e$ | Number of categorical attributes |
| $o_i$ | $i$-th object in a cluster |
| $o_i^c$ | Numeric values of $o_i$ |
| $o_i^e$ | Categorical values of $o_i$ |
| $X^r$ | Set of $o_i^r$ of a cluster $X$ |
| $X^c$ | Set of $o_i^c$ of a cluster $X$ |
| $X_i^c$ | Values of the $i$-th categorical attribute in $X^c$ |
| $H_{X, I}$ | Histogram of $X_i^c$ |
| $H_X$ | Histogram set of $X^c$ |
| $|X|$ | Cardinality of $X$ |

TABLE 3: Database example

| Cluster | Object | Age | Height (m) | Religion | Sex |
|---------|--------|-----|------------|----------|-----|
| $X$     | $o_1$  | 32  | 1.72       | Christianity | M   |
|         | $o_2$  | 34  | 1.63       | Christianity | F   |
|         | $o_3$  | 31  | 1.76       | Hinduism     | M   |
|         | $o_4$  | 41  | 1.66       | Christianity |     |
|         | $o_5$  | 29  | 1.80       | Islam        | M   |
|         | $o_6$  | 44  | 1.54       | Hinduism     | F   |
|         | $o_7$  | 21  | 1.71       | Hinduism     |     |
|         | $o_8$  | 23  | 1.65       | Islam        | F   |

Table 3 shows an example of a database having eight objects ($o_1, o_2, \ldots, o_8$) composed of both numeric and categorical attributes. In this example, we have two clusters $X = \{o_1, o_2, \ldots, o_8\}$ and $Y = \{o_6, o_7, o_8\}$. There are two numeric attributes, age and height, and two categorical attributes religion and sex. The values in the numeric attributes age and height are real numbers, whereas the values in the categorical attributes religion and sex are in $A_1 = \{\text{Christianity, Islam, Hinduism}\}$ and $A_2 = \{\text{M, F}\}$, respectively. We use the simple notations C, I, and H to represent Christianity, Islam, and Hinduism of the categorical attribute religion, respectively. For example, the 1st object $o_1$ is $(o_{1,1}^c, o_{1,1}^e, o_{1,1}^f)$ and $X^c = \{o_{1,1}^c, o_{1,2}^c, \ldots, o_{6,6}^c\}$. The cluster $X$ is composed of $X^r = \{o_{1,1}^r, o_{1,2}^r, \ldots, o_{6,6}^r\}$ and $X^c = \{o_{1,1}^c, o_{1,2}^c, \ldots, o_{6,6}^c\}$. Thus, $X_i^c = \{o_{1,1}^c, o_{1,2}^c, o_{1,3}^c, o_{1,4}^c, o_{1,5}^c\}$ and $X_i^r = \{o_{1,1}^r, o_{1,2}^r, o_{1,3}^r, o_{1,4}^r, o_{1,5}^r\} = \{C, C, H, C, I\}$. The histogram set of $X^c$ is $H_X = (H_{X, 1}, H_{X, 2})$, where $H_{X, 1} = \{(C, 3), (I, 1), (H, 1)\}$, and $H_{X, 2} = \{(M, 3), (F, 2)\}$.

B. MIXED-TYPE CLUSTERING FEATURES

We formally define the MCF vector, which is a quadruple maintaining summarized information about a cluster.

**Definition III.1.** MCF vector: Given $N$ $d$-dimensional objects in a cluster $X$: $\{o_i\}$, where $i = 1, 2, \ldots, N$, the MCF vector of $X$ is defined as $MCF_X = (N_X, L\bar{S}_X, SS_X, H_X)$, where $N_X, L\bar{S}_X$, and $SS_X$ are the same as $N$, $L\bar{S}$, and $SS$ in the CF vector in Definition II.1, and $H_X$ is a histogram set of $X^c$ of $X$.

The two histogram sets can be merged as follows.

**Definition III.2.** (Histogram set merge): Given two histogram sets $H_X$ and $H_Y$, the merged histogram set $H_{X \cup Y}$ of $H_X$ and $H_Y$ is obtained by the following formula:

$$H_{X \cup Y} = \bigcup_{i=1}^{d^e} (\zeta(H_{X, i}, H_{Y, i}))$$

, where

$$\zeta(H_{X, i}, H_{Y, i}) = \{(a_i : f_i) | a_i \text{ is a distinct value in } H_{X, i} \cup H_{Y, i}\}$$

and $f_i$ is the sum of $a_i$’s frequencies in $H_{X, i} \cup H_{Y, i}$.

For example, in Table 3, $\zeta(H_{X, 1}, H_{Y, 1}) = \{(\{C, 3\}, \{I, 1\}, \{H, 1\})\}$, $\{(I, 1), \{H, 2\}\} = \{(C, 3), \{I, 2\}, \{H, 3\}\}$. The time complexity of $H_{X, i} \cup H_{Y, i}$ depends on the data structures used to store the histogram. When the hash table is used to store $H_{X, i}$ and $H_{Y, i}$, the average time complexity of $H_{X, i} \cup H_{Y, i}$ is $O(A_i)$. Note that the average time complexity of the hash table’s search operation is $O(1)$.

The two clusters can be merged using their MCF vectors.

**Theorem III.1.** MCF Additivity Theorem: Let $MCF_X = (N_X, L\bar{S}_X, SS_X, H_X)$ and $MCF_Y = (N_Y, L\bar{S}_Y, SS_Y, H_Y)$ be the two MCF vectors of two clusters. The MCF vector of the cluster obtained by merging the two clusters is $MCF_{X \cup Y} = (N_X + N_Y, L\bar{S}_X + L\bar{S}_Y, SS_X + SS_Y, H_{X \cup Y})$.

**Proof of Theorem III.1.** The proof consists of straightforward algebra.

Theorem III.1 shows that an MCF vector holds the MCF additivity theorem, which is an extension of the CF additivity theorem [38]. Thus, the two clusters are merged in an accurate manner.

C. MEASURES FOR CLUSTERS

We compute the distance between two clusters as follows:

**Definition III.3.** Distance between clusters: Given $d^e$ numeric and $d^c$ categorical attributes, the distance between clusters $X$ and $Y$ is defined as

$$D(X, Y) = \gamma \cdot \frac{d_0(X^r, Y^r)}{\sqrt{d^e}} + (1 - \gamma) \cdot \frac{d_H(H_X, H_Y)}{d^c}$$

, where $d_0(X^r, Y^r)$ is the centroid Euclidean distance between $X^r$ and $Y^r$; $d_H(H_X, H_Y)$ is the distance between $X^c$ and $Y^c$, and $\gamma$ is the numeric attribute weight.

The distance between $X^r$ and $Y^r$ is obtained using the centroid Euclidean distance $d_0$, $d_H(H_X, H_Y)$ is the distance between $X^c$ and $Y^c$, which will be explained in Definition III.5 of Section III-D. Thus, we can compute the distance between the two clusters. The diameter of the cluster is also derived from Definitions II.2 and III.8, which will be given in Section III-D.
Definition III.4. Diameter of a cluster: The diameter of cluster $\mathcal{X}$ is defined as
\[ P(\mathcal{X}) = \gamma \frac{ar(X^c)}{\sqrt{d^c}} + (1-\gamma) \frac{\rho(HX)}{d^c}, \]
where $ar(X^c)$ is the average radius of $X^c$; $\rho(HX)$ is the diameter of $X^c$, and $\gamma$ is the numeric attribute weight.

The numeric attribute weight $\gamma$ is defined by an expert in the domain. It should be noted that we will demonstrate that $\gamma = 0.5$ is sufficient for our proposed distance measure. $\frac{1}{\sqrt{d^c}}$ and $\frac{1}{d^c}$ are used to normalize the distance and diameter of the numeric and categorical attribute measures, respectively.

D. CATEGORICAL DISTANCE MEASURE BASED ON HISTOGRAM

We present a distance measure based on the histogram of the MCF vector. This measure computes the distance between the categorical values of the two clusters using their histogram sets. Then, we show that this distance measure computes the same average distance as the average distance between objects of two clusters when only considering categorical attributes. In our distance measure, we use elements of the MCF vector $MCF_X$ of $X$, such as $H_X$ and $N_X$.

Definition III.5. Distance based on histogram: The distance between $H_X$ and $H_Y$ is obtained using the following formula:
\[ d_H(H_X, H_Y) = \frac{\omega(H_X, H_Y)}{N_X \cdot N_Y} \]

, where
\[ \omega(H_X, H_Y) = \sum_{i=1}^{d^c} \sum_{\langle a_j, f_j \rangle \in H_X, l_i} \sum_{l_i \neq a_k} f_j \cdot f_k. \]

We use Figure 3 to demonstrate the distance computation of the categorical attribute religion between $X^c$ and $Y^c$. In this figure, we use the simple notations $C$, $I$, and $H$ to represent Christianity, Islam, and Hinduism, respectively. Let $A_1 = \{C, I, H\}$. In Figure 3, two objects are connected by a solid line if they have the same value (the distance between them is zero); otherwise, two objects are connected by a dashed line (the distance between them is one). In Figure 3a, each element of the histogram $H_{X,1}$ forms a (value:frequency)-pair, i.e., (C:2). The distance between $H_{X,1}$ and $H_{Y,1}$ is
\[ \omega(H_{X,1}, H_{Y,1})/(N_X \cdot N_Y) = (3+1+3+2+4+1+2+1+1)/5 = 15/3. \]

Next, we introduce the distance between two values within the same categorical attribute. Using Definition III.6, we define the average distance between $X^c$ and $Y^c$.

Definition III.6. Given two categorical values $a_j, a_k \in A_l$, the distance between $a_j$ and $a_k$ is defined by the following formula:
\[ \delta(a_j, a_k) = \begin{cases} 0 & (a_j = a_k) \\ 1 & (a_j \neq a_k). \end{cases} \]

FIGURE 3: Distance between $X^c_1$ and $Y^c_1$ from Table 3

(a) Based on histogram
(b) Based on object

Definition III.7. Distance based on object: The distance between $X^c$ and $Y^c$ is obtained by the following formula:
\[ d(X^c, Y^c) = \frac{\sum_{i=1}^{d^c} \sum_{\langle a_j, f_j \rangle \in X^c} \sum_{\langle a_l, f_l \rangle \in Y^c} \delta(a_j, a_l) \cdot f_j \cdot f_l}{N_X \cdot N_Y}. \]

In Figure 3b, the distance between $X^c_1$ and $Y^c_1$ is (the number of dashed lines) $l$ (the number of all lines) $= 12/15$, which is the same as the distance obtained from the example in Figure 3a. This is because these two distance measures in Definitions III.5 and III.7 compute the same distance. Now, we prove that the distance based on the histogram and the distance based on the object are the same.

Theorem III.2. Given $X^c$ and $Y^c$, the following equation holds:
\[ d(X^c, Y^c) = d_H(H_X, H_Y). \]

Proof of Theorem III.2. Let $\beta^c = (\beta^c_1, \beta^c_2, \ldots, \beta^c_6)$ be a list of the categorical values of a mixed-type object $\beta$, and $X^c = \{\alpha^c_1, \alpha^c_2, \ldots, \alpha^c_5\}$. The distance between $X^c$ and $\beta^c$ is defined [19] as follows:
\[ \text{dist}(X^c, \beta^c) = \frac{\sum_{i=1}^{d^c} \sum_{\langle a_j, f_j \rangle \in X^c} \delta(a_j, \beta^c_i) \cdot f_j \cdot f_l}{N}. \]

In other words, the sum of frequencies in $H_{X,1}$, whose categorical values are different from $\beta^c_i$, is $\phi(H_{X,1}, \beta^c_i)$. Therefore, Eq. (7) can be replaced with
\[ \phi(H_{X,1}, \beta^c_i) = \sum_{\langle a_j, f_j \rangle \in H_{X,1}, \langle a_j, f_j \rangle \in H_{Y,1}, a_j \neq \beta^c_i} f_j \cdot f_l. \]

If $\beta^c_i$ is changed with the histogram $H_{Y,1}$ of $Y^c$, in Eq. (8),
\[ \phi(H_{X,1}, H_{Y,1}) = \sum_{a_j \neq \beta^c} f_j \cdot f_l. \]
Using Eq. (9), Eq. (6) is extended to obtain the average distance between $X^c$ and $Y^c$:

$$d_{H}(X^c, Y^c) = \frac{\omega(H_X, H_Y)}{N_X \cdot N_Y} \sum_{i=1}^{d^c} \sum_{(a_i,j_i) \in H_{X,i}} \sum_{(a_k,k) \in H_{Y,j}, a_k \neq a_j} f_j \cdot f_k = \frac{\omega(H_X, H_Y)}{N_X \cdot N_Y} \sum_{i=1}^{d^c} \phi(H_{X,i}, H_{Y,i}) = \frac{\omega(H_X, H_Y)}{N_X \cdot N_Y} \sum_{i=1}^{d^c} \sum_{o_i \in X^c} \sum_{o_j \in Y^c} \delta(o^c_{i,i}, o^c_{j,j}) \leq d(X^c, Y^c).$$

The distance based on the histogram decreases the computational cost and memory usage because it takes less time and space complexities. Given $X^c$ and $Y^c$, the distance computation between $X^c$ and $Y^c$ based on the object takes $O(|X^c| \cdot |Y^c|)$. By contrast, the distance computation between $H_X$ and $H_Y$ takes $O(|H_X| \cdot |H_Y|)$. Because $|X^c| \geq |H_X|$ and $|Y^c| \geq |H_Y|$, $O(|X^c| \cdot |Y^c|) \geq O(|H_X| \cdot |H_Y|)$ always holds. For the same reason, maintaining $H_X(H_Y)$ takes less space than $X^c(Y^c)$ in most cases.

Because histograms can contain only one object, they are easily applied to a distance computation between a cluster and an object or between two objects. The average distance of all object pairs in $X^c$ shows the closeness of all objects of $X$ with regard to the categorical values. We refer to this in the terms of the “diameter” of $X^c$.

**Definition III.8. Diameter:** The diameter $\rho(H_X)$ is obtained using the following formula:

$$\rho(H_X) = \begin{cases} \frac{\psi_{H_X}}{N_X(N_X-1)}, & \text{if } N_X \geq 2 \\ 0, & \text{otherwise.} \end{cases}$$

, where

$$\psi_{H_X} = \sum_{i=1}^{d^c} \sum_{(a_i,j_i) \in H_{X,i}} f_j \cdot (N_X - f_j).$$

In the examples in Figure 4, the symbolic notations and line types are the same as those in Figure 3. The diameter of $H_{X,1}$ in Figure 4a, which is $\rho(H_{X,1}) = (3 \cdot 2 + 1 \cdot 4 + 1 \cdot 4) / (5 \cdot 4) = 7/10$, is the same as the diameter of $X^c_1$ in Figure 4b, which is (the number of dashed lines) / (the number of all lines) = 7/10.

**IV. MCF TREE**

We describe the structure and building process of the MCF tree. We also propose a tree-rebuilding process based on an adaptive memory-size-increasing scheme to estimate a more accurate threshold for building an MCF tree. Furthermore, we explain how to apply the range-query-based clustering method ERC to the MCF tree.

**A. STRUCTURE AND INSERTION PROCESS OF MCF TREE**

The MCF tree is a height-balanced tree. It consists of the MCF vectors to summarize a mixed-type data set. A cluster is stored and maintained in the form of an MCF vector. The distance between clusters is computed using their MCF vectors. Two clusters can be merged by the MCF additivity theorem, similar to a CF+ tree. However, the structure and building process of the MCF tree are slightly different from the CF+ tree because it requires the minimum number of entries of each node. We first extend the definition of the CF+ tree to the MCF tree and explain its building process.

**Definition IV.1. The MCF tree satisfies the following properties:**

1) Each non-leaf node contains, at most, $C$ entries in the form $[MCF_i, \text{child}_i]$, where $i = 1, 2, \ldots, C$; child$_i$ is a pointer to its $i$-th child node, and MCF$_i$ is the MCF vector of the subcluster pointed to by child$_i$. An entry of the non-leaf node is called a subcluster.

2) Each leaf node contains, at most, $L$ entries in the form $[MCF_i]$, where $i = 1, 2, \ldots, L$. A entry of the leaf node is called a microcluster.

3) All leaf nodes are chained together for efficient sequential scanning using the “prev” and “next” pointers.

4) Each node has at least $\phi$ entries of the maximum number of entries (i.e., $L$ or $C$).

The building process of an MCF tree is the same as that of the CF+ tree. The only difference is the adaptable node split scheme of the MCF tree. It divides all entries of the node into two entry sets until each entry set has at least $\phi$ entries of the maximum number of entries of the node. Once the MCF tree is built, existing clustering methods, such as ERC and W-K Prototypes, group a set of microclusters of the MCF tree into the final clusters.

**B. TREE BUILDING PROCESS BASED ON ADAPTIVE MEMORY-SIZE-INCREASING SCHEME**

In CF+ - ERC, when a given memory size is too small, the objects from the different clusters are merged into the same
microcluster. This implies that the accuracy of global clustering using these microclusters is degraded. On the contrary, if a given memory size is too large and the initial threshold is too small, the CF tree built using the threshold estimation scheme does not summarize the data set effectively. This is because a small threshold rarely absorbs the objects into the same microcluster. However, the estimation of an accurate memory size for the data sets is a difficult problem because it differs depending on the characteristics of the data sets. Therefore, we propose an adaptive memory-size-increasing scheme that estimates a more accurate threshold by adaptively increasing memory size according to the amount of size reduction of the re-built MCF tree.

We introduce the tree building process and then explain our threshold estimation based on the adaptive memory-size-increasing scheme in detail. Our threshold estimation scheme has three functions: initial memory-size-estimation, threshold increase, and memory-size-increase. The MCF tree is built as follows: Let \( S \) be a data set composed of \( d' \) numeric and \( d'' \) categorical attributes, \( MM \) be the main memory size, and \( M(W) \) be the memory size required to store \( W \).

1. \( i = 0 \).
2. Compute a small initial memory size \( \mu_i \) (the initial memory-size-estimation function).
3. If \( \mu_i \geq MM \), then \( \mu_i = MM \).
4. Create an MCF tree \( Tr_i \) with a threshold \( T_i = 0 \).
5. Loop until all the objects of the data set \( S \) are read:
   a) Read an object \( o \) from \( S \) and insert \( o \) into \( Tr_i \).
   b) If \( M(Tr_i) > \mu_i \),
      i) \( \alpha = M(Tr_i) \) and \( i = i + 1 \).
      ii) Compute the \( i \)-th increased threshold \( T_i \) (the threshold increase function).
      iii) If \( T_i \), then \( T_i = T_{i-1} \times 2 \).
      iv) Rebuild the \( i \)-th MCF tree \( Tr_i \) from \( Tr_{i-1} \) by using \( T_i \).
      v) \( \beta = M(Tr_i) \).
      vi) Increase \( \mu_{i-1} \) to \( \mu_i \) (the memory-size-increase function using \( \alpha \) and \( \beta \)).
      vii) If \( \mu_i \geq MM \), then \( \mu_i = MM \).

The steps i) – vii) are the tree rebuilding process. These steps are performed repeatedly until all the objects are inserted into the MCF tree. If the memory size for the \( i \)-th MCF tree, \( M(Tr_i) \), is greater than the main memory size \( MM \), the main memory size is used as the memory size, and the memory size does not thereafter increase. The two variables \( \alpha \) and \( \beta \) store the memory sizes of the two MCF trees immediately before and immediately after rebuilding the MCF tree, which will be used in the memory-size-increase function.

In the initial memory-size-estimation function, a small initial memory size \( \mu_0 \) is roughly computed using the MCF tree \( Tr_0 \), consisting of \( \pi \times |S| \) objects for the first tree rebuilding. All entries of the nodes in the MCF tree are represented by MCF vectors. Therefore, we need to obtain the number of entries of \( Tr_0 \) and the memory size of the MCF vector.

Let \( B \) be the branching factor of \( Tr_0 \). Let \( a_i \) denote the number of entries of all nodes whose heights are \( i \). The number of entries of all leaf nodes (i.e., their heights are 0) is the same as the number of objects. The reason is that each entry, for all leaf nodes of \( Tr_0 \) built by using \( T_0 = 0 \), contains one object. Thus, \( a_0 = \frac{|S|}{\pi} \). In \( Tr_0 \), the non-leaf node at each level has up to \( B \) entries. Hence, the total number of entries in the MCF tree can be approximately computed as

\[
\kappa = \sum_{i=0}^{n} a_i,
\]

where \( h \) is the number of recursions until \( a_1 = 0 \). Because the histogram of each categorical attribute of the microcluster \( \chi \) contains at least one \((key : value)\)-pair, its minimum memory size is \( M(H_X) = (M(key) + M(value)) \times d^c \).

Thus, \( M(MCF_X) \) of the MCF vector \( MCF_X \) is obtained by the following formula:

\[
M(MCF_X) = M(N_X) + M(LS_X) + M(SS_X) + M(H_X).
\]

From \( \kappa \) and \( M(MCF_X) \), the small initial memory size is computed as

\[
\mu_0 = M(MCF_X) \times \kappa.
\]

In the threshold increase function, the \( i \)-th estimated threshold \( T_i \) is increased from \( T_{i-1} \) using the \((i-1)\)-th rebuilt MCF tree \( Tr_{i-1} \). We use the term “leaf minimum distance” of a leaf node to refer to the minimum distance among the microclusters belonging to the leaf node. While building \( Tr_{i-1} \), it tends to assign similar microclusters to the same leaf node. Therefore, we can use all the leaf minimum distances to compute a new threshold \( T_i \) to reduce the computational cost, instead of using the minimum distance between all microclusters pairs of the tree. Because the minimum distance of the leaf node depends on the microclusters assigned to the leaf node, some leaf minimum distances might be much larger than the average of the leaf minimum distances. We exclude such leaf minimum distances to obtain a newly increased threshold, as follows.

**Definition IV.2.** Threshold increase: Given a set \( \rho \) of the leaf minimum distances of all leaf nodes in the \((i-1)\)-th rebuilt MCF tree \( Tr_{i-1} \), with the \((i-1)\)-th increased threshold \( T_{i-1} \), we let \( \bar{\rho} \) and \( \sigma \) denote the average and standard deviation of \( \rho \) respectively. The set \( \omega \) of leaf minimum distances is

\[
\omega = \{ \rho_i | \rho_i \in \rho \land \rho_i < \bar{\rho} + \sigma \}.
\]

Let \( \bar{\omega} \) be the average of \( \omega \). If \( \bar{\omega} < T_{i-1} \), then \( T_i = T_{i-1} \). Otherwise, \( T_i = \bar{\omega} \).

The memory-size-increase function increases the memory size adaptively. By Definition IV.2, the estimated threshold increases monotonically as the number of tree rebuilds increases. Therefore, we use the number of tree rebuilds to estimate a more accurate threshold through the memory size reduction ratio (i.e., \( \frac{\mu}{\kappa} \)). If \( \bar{\omega} < T_{i-1} \), the memory size is increased by the previous increment in memory size i.e.,
Increasing scheme. In Figure 5a, the memory size of the MCF tree is increased by increasing the memory size estimation. Consequently, we estimate the more conservative threshold estimation becomes more aggressive. Thus, $T_i$ is set to zero when using the same memory size. This indicates that the estimation of $T_i$ is rather aggressive. Thus, we increase the memory size significantly to reduce the number of tree rebuilds. As a result, the threshold increases are estimated less frequently, which makes the current aggressive threshold estimation more conservative. Consequently, we estimate the more accurate threshold by increasing the memory size adaptively while building the MCF tree.

Figure 5 shows an example of the adaptive memory-size-increasing scheme. In Figure 5a, the memory size $M(T_{r1})$ of the MCF tree $T_{r1}$ built using $T_1 = 4$ becomes larger than the 1st increased memory size $\mu_1$. At that time, $\alpha$ is set to $M(T_{r1})$. Because $T_{r1}$ consists of four leaf nodes $l_{n1}, l_{n2}, l_{n3}$, and $l_{n4}$, we compute their leaf minimum distances $\rho_1 = 5, \rho_2 = 6, \rho_3 = 4$, and $\rho_4 = 9$, its average distance $\bar{\rho} = 6$, and the standard deviation $\sigma = 1.9$. Because $\rho_4 \geq \bar{\rho} + \sigma$ and $\bar{\rho} \geq T_1$, $T_{r1}$ is set to 5. Next, $T_{r2}$ is rebuilt using $T_2$. It should be noted that $M(T_{r2})$ is always smaller than $M(T_{r1})$ when $T_2$ is greater than $T_1$. Then, $\beta$ is set to $M(T_{r2})$, as shown in Figure 5b. The memory size is adaptively increased from $\mu_1$ to $\mu_2$ using $\alpha$ and $\beta$, as shown in Figure 5c. The remaining objects of the data sets are inserted into $T_{r2}$ based on $T_2$, until $M(T_{r2})$ is greater than $\mu_2$.

C. APPLICATION OF ERC TO MCF TREE

ERC has been proposed as a global clustering method [35]. It reduces the clustering time by using the CF+ tree structure. In this section, we explain how to apply the ERC method to the MCF tree. The distance measure $d_{0}$ and average distance $ar$ of the ERC method handle only the numeric values. Thus, $d_{0}$ and $ar$ are replaced with our distance measure $D$ and diameter measure $P$, respectively. The three definitions and one method (ERQ) of ERC, which use $d_{0}$ and $ar$, are modified to use $D$ and $P$. Note that the ERQ method is called by the ERS method of ERC.

First, we formally change $IMD$ to $IMD_M$ for mixed-type objects.

Definition IV.3. Inter-microcluster distance: Given two microclusters $\mathcal{X}$ and $\mathcal{Y}$ with their corresponding diameters $P(\mathcal{X})$ and $P(\mathcal{Y})$, the inter-microcluster distance between $\mathcal{X}$ and $\mathcal{Y}$ is defined as

$$IMD_M(\mathcal{X}, \mathcal{Y}) = D(\mathcal{X}, \mathcal{Y}) - (P(\mathcal{X}) + P(\mathcal{Y})).$$

The partition step of the ERC method divides a set of microclusters into a set of MCSs using $IMD_M$. Additionally, the ERQ method of the refinement step of the ERC method uses $IMD_M$ to find out connections between MCSs. Subsequently, the radius of each MCS is obtained as follows:

Definition IV.4. Radius of MCS: Let $\mathcal{MCS}_i$ be an MCS consisting of $k$ microclusters $\mathcal{MC}_1, \mathcal{MC}_2, \ldots, \mathcal{MC}_k$. Their diameters are $P(\mathcal{MC}_1), P(\mathcal{MC}_2), \ldots, P(\mathcal{MC}_k)$. Then, for $\mathcal{MC}_i$, the radius $\mathcal{MCS}_i.r$ is defined as $MCS_i.r = \max_{1 \leq y \leq k} \{D(\mathcal{MC}_y, \mathcal{MC}_i) + P(\mathcal{MC}_y)\}$.

The ERC method performs a range query using the CF+ tree structure. To perform the range query, the subcluster must have a radius that covers all its descendant microclusters. Similar to the CF+ tree, the radius of the subcluster of the MCF tree is formally defined as follows:

Definition IV.5. Radius of subcluster: On a non-leaf node in an MCF tree, the $i$-th subcluster $\mathcal{SC}_i$ consists of $m$ clusters $C_1, C_2, \ldots, C_m$. The radius $\mathcal{SC}_i.r$ is defined as

$$\mathcal{SC}_i.r = \begin{cases} \max_{1 \leq j \leq m} \{D(C_i, C_j) + P(C_j)\}, & \text{if } C_j \text{ is a leaf (microcluster)} \\ \max_{1 \leq j \leq m} \{D(C_i, C_j) + C_j.r\}, & \text{otherwise}. \end{cases}$$

Because the subcluster can have child subclusters, the radius of the subcluster can be computed recursively. For example, when the height of the MCF tree is three and...
the subcluster \( SC_r \) of the root node has \( m \) child subclusters \( SC_i (1 \leq i \leq m) \), each child subcluster has \( l \) microclusters \( C_j (1 \leq j \leq l) \). Thus, the radius of \( SC_r \) is \( \max_{1 \leq i \leq m} \{ d( SC_r, SC_i ) + \max_{1 \leq j \leq l} \{ d( SC_i, C_j ) + P( C_j ) \} \} \).

The radius of a subcluster must cover all its descendant microclusters. In other words, the distance between the subcluster and all microclusters must be less than the radius. Assume that the radius of \( SC_r \) is the sum of the three distances \( d( SC_r, SC_i ) + d( SC_i, C_j ) + P( C_j ) \). If \( d( SC_r, SC_i ) + P( C_j ) \leq d( SC_r, SC_i ) + d( SC_i, C_j ) + P( C_j ) \), \( SC_r \) must cover all its descendant microclusters. The above inequality can be simplified as \( d( SC_r, SC_i ) \leq d( SC_r, SC_i ) + d( SC_i, C_j ) \).

The distance measure \( d_0 \) holds the inequality because it uses the Euclidean distance. Hence, we only show that the distance measure \( d_0 \) also holds the inequality.

**Theorem IV.1.** Triangle inequality: Given \( X^c \), \( Y^c \), and \( Z^c \), the following inequality holds:

\[
d( X^c, Z^c ) \leq d( X^c, Y^c ) + d( Y^c, Z^c ). \tag{10}
\]

**Proof of Theorem IV.1.** Let \( X^c = ( o_{1}^{c}, o_{2}^{c}, \cdots, o_{f}^{c} ) \), \( Y^c = ( o_{i}^{c}, o_{2}^{c}, \cdots, o_{g}^{c} ) \), and \( Z^c = ( o_{i}^{c}, o_{2}^{c}, \cdots, o_{g}^{c} ) \) be the Hamming distance between \( o_{i}^{c} \) and \( o_{j}^{c} \).

\[
d( X^c, Z^c ) \leq d( X^c, Y^c ) + d( Y^c, Z^c ) \leq \frac{\eta( X^c, Y^c )}{e \cdot g} + \frac{\eta( Y^c, Z^c )}{f \cdot g}, \tag{11}
\]

where

\[
\eta( X^c, Z^c ) = \sum_{x_{i}^{c} \in X^c} \sum_{z_{k}^{c} \in Z^c} HD( x_{i}^{c}, z_{k}^{c} ).
\]

We rewrite each term of inequation (11) as follows:

\[
f \cdot \eta( X^c, Z^c ) = f \cdot \eta( X^c, Z^c ) \leq f \cdot \eta( X^c, Z^c ) + f \cdot \eta( X^c, Z^c ) + \cdots + f \cdot \eta( X^c, Z^c ) \leq f \cdot \eta( X^c, Z^c ) \leq \eta( X^c, Z^c ) + \eta( X^c, Z^c ) + \cdots + \eta( X^c, Z^c ) = \eta( X^c, Z^c ).
\]

Therefore, inequation (11) is rewritten as follows:

\[
\sum_{i=1}^{e} \sum_{j=1}^{f} \sum_{k=1}^{g} HD( x_{i}^{c}, z_{k}^{c} ) + \sum_{i=1}^{e} \sum_{j=1}^{f} \sum_{k=1}^{g} HD( y_{j}^{c}, z_{k}^{c} ) \leq \sum_{i=1}^{e} \sum_{j=1}^{f} \sum_{k=1}^{g} HD( x_{i}^{c}, y_{j}^{c} ) + \sum_{i=1}^{e} \sum_{j=1}^{f} \sum_{k=1}^{g} HD( y_{j}^{c}, z_{k}^{c} ) \tag{12}
\]

For all \( i \in [1, e], h \in [1, f] \), \( \sum_{j=1}^{f} HD( x_{i}^{c}, z_{k}^{c} ) \), \( \sum_{j=1}^{f} HD( x_{i}^{c}, y_{j}^{c} ) \), and \( \sum_{j=1}^{f} HD( y_{j}^{c}, z_{k}^{c} ) \) are divided into \( f \) inequalities as follows:

\[
\begin{align*}
HD( x_{i}^{c}, z_{k}^{c} )_1 & \leq HD( x_{i}^{c}, y_{j}^{c} )_1 + HD( y_{j}^{c}, z_{k}^{c} )_1, \\
HD( x_{i}^{c}, z_{k}^{c} )_2 & \leq HD( x_{i}^{c}, y_{j}^{c} )_2 + HD( y_{j}^{c}, z_{k}^{c} )_2, \\
& \quad \vdots \\
HD( x_{i}^{c}, z_{k}^{c} )_f & \leq HD( x_{i}^{c}, y_{j}^{c} )_f + HD( y_{j}^{c}, z_{k}^{c} )_f.
\end{align*} \tag{13}
\]

Because the Hamming distance holds the triangle inequality, all inequalities in inequation (13) also hold. Therefore, \( d( X^c, Z^c ) \leq d( X^c, Y^c ) + d( Y^c, Z^c ) \).

Because \( d \) and \( d_H \) are the same, it is guaranteed that the radius of the subcluster of the MCF tree covers its descendant microclusters in accordance with Theorem IV.1. This indicates that the range query based on the radii of the MCF tree determines the microclusters exactly.

**V. PERFORMANCE ANALYSIS**

In this section, we provide theoretical and experimental analyses of our proposed clustering method.

**A. THEORETICAL ANALYSIS**

We analyze the time complexities of our proposed clustering method. Let \( N \) be the number of data objects, \( B \) be the branching factor of the MCF tree, \( d^r \) be the number of numeric attributes, and \( d^c \) be the number of categorical attributes. We assume that each leaf entry of the MCF tree absorbs \( a \) data objects on average. Therefore, let \( n = \frac{N}{a} \) be the number of microclusters.

We analyze the time complexity of building an MCF tree. An object is inserted into the MCF tree based on the closest criteria on a node with up to \( B \) entries. The time complexity of the insertion of the object is \( O( B \cdot d \cdot \log_B n ) \), where \( \log_B n \) is the height of the MCF tree consisting of \( N \) objects. Thus, the time complexity of constructing a complete MCF tree for the entire data set is \( O( N \cdot B \cdot d \cdot \log_B n ) \).

When rebuilding the MCF tree, inserting microclusters into the MCF tree takes \( O( n \cdot B \cdot d \cdot \log_B n ) \). The number of rebuilds depends on \( \pi \) of our adaptive memory-size-increasing scheme. Therefore, rebuilding the MCF tree takes \( O( \frac{N}{a} \cdot B \cdot d \cdot \log_B n ) \).

However, because the estimated threshold monotonically increases as the number of tree rebuilds increases, more objects are merged into the same microcluster every time the MCF tree is rebuilt. This, in turn, reduces both the number of microclusters and the height of the MCF tree. As a result, an increase in the number of rebuilds of the MCF tree reduces the time complexity of our proposed method.

**B. EXPERIMENTAL SETUP**

The performance of our method is compared with five existing clustering methods using various synthetic data sets and different real data sets. All experiments were carried out on a computer with an eight-core 2.5-GHz CPU, 32 GB of RAM,
and a 1-TB SSD. In addition, we set $C = 10$ and $L = 10$ for the MCF tree. In the MCF tree, we keep at least 20% of the entries in each node to distribute the entries more evenly during the node split.

1) Comparative Methods
Ahmad and Dey [5], convex K-means [32], W-K-prototypes [24], the extension of W-K-prototypes [27], and K-mixed prototypes [10] were used as the comparative non-summary-based clustering methods. These are referred to as AhmadK, ConvexK, WKP, JIWKP, and KMP, respectively.

Our proposed MCF tree-based clustering method uses not only ERC but also the non-summary-based clustering methods as the second-phase global clustering scheme. For example, the method that uses WKP as the global clustering method is called MCF-WKP. Because they require mixed-type input data, all microclusters must be converted into objects in a mixed-type form. For each microcluster, the centroid (mode) of each numeric (categorical) attribute is used as the numeric (categorical) value of the converted object. Subsequently, global clustering methods are used to group sets of converted objects into the final clusters.

We also compared our method to the summary-based clustering method that Chiu et al. [13] had proposed. It used a CF tree for clustering the mixed-type data sets by using the modified CF vectors. To distinguish the tree from the CF tree of BIRCH, we used the term “CF$^M$ tree” to refer to the CF tree. Similar to our method, the non-summary-based clustering methods were used as a global clustering method of the CF$^M$-based clustering method.

In the experiments, a random centroid initialization scheme was used for all K-means extension methods. For all K-means extension methods, we set the maximum number of iterations to 100. This is because almost all experiments were performed before 100 iterations, or there were no significant performance improvements thereafter.

2) Validation Measure
The average purity, inverse purity [7], Rand index [34], and execution time of these clustering methods were used as the primary performance measures. They were obtained by repeating the experiment ten times with different seeds on the same data set. The purity and inverse purity measures were formally described in [7]. The purity of the clustering method evaluates the frequency of the most similar objects in each cluster, while the inverse purity of the clustering method evaluates similar objects placed in the same cluster. The execution time includes the building time of the MCF and CF$^M$ trees.

3) Data Sets
Diverse synthetic and real data sets were used in our experiments. Synthetic data sets were obtained using our synthetic data generator based on the parameters shown in Table 4.

All the synthetic data sets consist of $K$ clusters with $d^e$ numeric and $d^c$ categorical attributes. All numeric attributes take real values, and all categorical attributes represent one of the distinct values. All attributes are assumed to be independent. The numeric attributes are generated based on the normal distributions and the categorical attributes are generated based on the multinomial distributions. The cluster centers are randomly located in the mixed-type data space and the radius of each cluster is randomly selected from $r_{min}$ to $r_{max}$. The number of objects in a cluster depends on the density of the cluster. All objects in each cluster are randomly generated around the cluster center within the radius.

We deliberately doubled $K$ and halved $N$ to differ the average density of all clusters in all data sets. Because other parameters were not changed, varying $N$ and $K$ changed the density of every cluster. In our experiment, $K$ doubled from 4 to 8 to 16 · · · to 512 to 1024, when $N$ was halved from 25000 to 12500 to 6250 · · · to 195 to 98. All object counts of the data set were randomly placed in the mixed-type data space. Nine data sets were generated by the synthetic data generator. The numbers of objects in these data sets were 233 k, 40 k, 179 k, 133 k, 158 k, 137 k, 141 k, 149 k, and 145 k. The default data set had the parameter values shown in Table 5.

The cover type data set (CovType), the CovPokElec data set (CPE), and the KDDCup99 data set (KDD99) were used. All data sets were provided by OpenML1. The KDD99 data set was used to evaluate clustering methods in previous experiments [11], [12], [16]. All numeric attributes of all data sets were normalized [5]. In experiments using ConvexK, each $q$-ary categorical attribute was converted to a 1-in-$q$ representation [32]. The weights of ConvexK and the $\beta$ of WKP and JIWKP were experimentally obtained. The experimental

| Parameter | Value or Range |
|-----------|----------------|
| Minimum radius range of cluster $r_{min}$ | 1 |
| Maximum radius range of cluster $r_{max}$ | 3 |
| Number of categorical attributes: $d^c$ | 4 |
| Number of numeric attributes: $d^e$ | 16 |
| Number of clusters: $K$ | 4, 8, · · · , 1024 |
| Average number of objects in a cluster: $N$ | 25000, 12500, · · · , 98 |

| Data set | $d^e$ | $d^c$ | $K$ | $N$ | $n$ | $\beta$ | weights |
|----------|-------|-------|-----|-----|-----|--------|---------|
| DDS | 1 | 3 | 4 | 16 | 256 | 391 | 141745 |
| CovType | 10 | 44 | 7 | 581012 | 8 | 0.03, 0.97 |
| CPE | 15 | 57 | 10 | 1455525 | 8 | 0.44, 0.56 |
| KDD99 | 34 | 7 | 23 | 4898431 | 10 | 0.27, 0.73 |


1https://www.openml.org/
parameters for these data sets are listed in Table 6.

C. EXPERIMENTAL ANALYSIS

In this section, we discuss the effects of the parameters on our proposed method, such as the $\pi$ of the adaptive memory-size-increasing scheme and the numeric attribute weight $\gamma$ of our distance measure. We also show the performance of our proposed method using three real data sets compared to the other clustering methods.

1) Effect of Adaptive Memory-Size-Increasing Scheme

The effect of the adaptive memory-size-increasing scheme on the performance of our clustering method was analyzed using nine synthetic mixed-type data sets. In this experiment, we increased the percentage $\pi$ from 1% to 5% and the number $K$ of clusters from 4 to 1024. Note that our threshold scheme dynamically increases the memory size. We set the numeric attribute weight $\gamma$ of our distance measure to 0.5.

The performance of our method was analyzed using five non-summary-based clustering methods. Because they all showed the same performance trend in terms of purity, inverse purity, and execution time, we will only discuss MCF-WKP and MCF-ERC in this study.

In Figures 6-7, with an increase in $\pi$ for each $K$, the execution times of all methods increased, whereas their purities and inverse purities were similar, except for $K = 512$ and $K = 1024$. Note that the numbers of microclusters in Figures 6c and 7c were identical because these two MCF trees were built by using the same data sets and parameters (i.e., $\pi$ and $\gamma$). Our proposed method found a group consisting of similar objects and then summarized it in a microcluster. This indicates the number of microclusters was much fewer than the number of objects for each data set as shown in Figure 6c. Therefore, the clustering time of our proposed method was reduced while maintaining its accuracy.

However, in Figures 6a-7a, the MCF tree excessively summarized the data sets with $K = 512$ and $K = 1024$. This is because our proposed method with a low $\pi$ obtained a large initial threshold; hence, objects of different true clusters were summarized in the same microcluster. When each cluster had a small number of objects, and the number of clusters was relatively high, our proposed method computed an overestimated initial memory size using $\pi$ objects, which were sparsely distributed in the data space. For these data sets whose objects, the MCF-based clustering methods with $\pi = 3\%$ show similar purity and inverse purity to those of other methods, as shown in Figures 6-7. According to these experiments, the proper $\pi$ of our adaptive memory-size-increasing scheme is proportional to $\frac{\pi}{\sqrt{n}}$, where $n$ is the number of objects and $K$ is the expected number of clusters.

We consider that experts in the domain can predict the desired number of clusters. In the real data sets, the numbers of clusters were usually less than those of these synthetic data sets while containing more objects. For the real data sets,
FIGURE 7: Effect of $K$ and $\pi$ on MCF-ERC

(a) Purity

(b) Inverse Purity

(c) Number of Microclusters

(d) Time

FIGURE 8: Effect of $K$ on clustering methods

(a) Purity

(b) Inverse Purity

(c) Number of Data

(d) Time
we found that $\pi = 0.06\%$ was sufficient for our adaptive memory-size-increasing scheme.

Next, we compared our summary-based clustering method to the non-summary-based clustering method WKP to demonstrate the efficiency of our proposed approach using the nine synthetic mixed-type data sets. We set $\pi = 3\%$ for all data sets because it was sufficient to cluster all synthetic data sets as shown in Figures 6-7.

Figure 8 shows that the MCF-based clustering methods, such as MCF-WKP and MCF-ERC, clustered these data sets more rapidly than WKP owing to fewer microclusters than objects in each data set. Therefore, our proposed method enhances the efficiency of clustering methods while maintaining clustering accuracy in terms of purity and inverse purity.

Now, we show how the memory size and threshold in our adaptive memory-size-increasing scheme were increased as the number of rebuilds of the MCF tree is progressed in Figure 9. Two variables $\alpha$ and $\beta$ were set to the memory sizes of the two MCF trees immediately before and immediately after rebuilding the MCF tree. A variable $\mu$ represents the increased memory size.

In Figure 9, the small initial memory sizes of DDS and KDD99 (i.e., $\pi = 1\%$ for DDS and $\pi = 0.06\%$ for KDD99) were adaptively increased every time the MCF trees were rebuilt. In Figure 9a, the threshold was increased until the $3^{rd}$ rebuild of the MCF tree and then it remained the same afterward. The reason was that the threshold estimated at the $3^{rd}$ and later rebuilds of the MCF tree was lower than that at the $2^{nd}$ rebuild of the MCF tree. In BIRCH, the threshold was doubled when the newly computed threshold was lower than the current threshold. As a result, if the threshold increased repeatedly like BIRCH, it is likely to be over-estimated.

In Figure 9b, $\beta$ was very smaller than $\alpha$ at the $1^{st}$ rebuild of the MCF tree, which indicates the threshold was increased too aggressively. Thus, the memory size $\mu$ was significantly increased to reduce the frequency of the tree rebuild. Similarly, the memory size was largely increased in the $2^{nd}$, $5^{th}$ rebuilds of the MCF tree because the newly increased thresholds were sufficiently large to summarize the KDD99 data set. Consequently, our adaptive memory-size-increasing scheme estimated the proper memory size for each data set without increasing the memory size too conservatively or aggressively.

2) Effect of Numeric Attribute Weight
The effect of the numeric attribute weight $\gamma$ of our distance measure on the performance of our clustering method was analyzed using DDS and KDD99. We set $\pi = 1\%$ for DDS and $\pi = 0.06\%$ for KDD99.

Figure 10 shows that the purity and inverse purity of all methods were similar when varying $\gamma$ for the synthetic data set. In Figure 11, when varying $\gamma$, the purity and inverse purity of MCF-ERC significantly decreased for the real data set, except for $\gamma$s between 0.4 and 0.5. This indicates that the balanced attribute weight was suitable for summarizing the synthetic and real data sets. This is because our method with extreme numeric attribute weights, such as $\gamma = 0.1$ (0.9), rarely used the numeric (categorical) values of the mixed-type data sets for clustering, which decreased the clustering accuracy, as shown in Figure 11b. Therefore, we set $\gamma = 0.5$ in all the subsequent experiments.

3) Effects for Real Mixed-type Data Sets
The performance of our method was analyzed by comparing non-summary-based clustering methods and CF$^M$-based clustering methods using three real mixed-type data sets. The non-summary-based clustering methods include AhmadK [5], ConvexK [32], WKP [24], JIWKP [27], and KMP [10]. We set the memory size of CF$^M$-based clustering methods to 64 MB that was the same memory size in the experiments [13]. We also set the memory size of the CF$^M$-based clustering methods to 32 MB and 128 MB for showing the effect of varying the memory size. CF$^M$-based clustering method is represented by its global clustering method and memory size. For example, CF$^M$-WKP (32M) is CF$^M$-based clustering method that uses WKP as the global clustering method and uses 32 MB memory size for building CF$^M$ tree. We set $\pi = 0.06\%$ for all real data sets. The results of these experiments are shown in Table 7.
In Table 7, the comparative non-summary-based clustering methods and MCF-tree-based clustering methods usually demonstrate similar performance in terms of purity, inverse purity, and Rand index. Because the MCF tree gathered similar objects into the same microcluster, microclusters can be used to cluster a mixed-type data set. Because the number of input objects has a significant impact on the time complexity of most clustering methods, including the methods used in our experiments, our method can cluster more rapidly than diverse non-summary-based clustering methods.

Table 7 shows that the inverse purities of our methods were higher than those of non-summary-based clustering methods. The reason was that the MCF tree gathered similar objects into the same microcluster, and these similar objects were not assigned to the different final clusters by the global clustering methods. Non-summary-based methods occasionally assigned similar objects into different final clusters, which dwindled the inverse purity. This is one of the benefits of the summary-based clustering method, as opposed to some clustering methods, such as K-means, that only consider the distance between centroids and objects, which may cause similar objects to be grouped into different final clusters.

The CPE and KDD99 data sets were relatively large compared to the CovType data set, and the execution times of our methods using these data sets were significantly smaller than those of non-summary-based clustering methods. The main reason for the reduced clustering time was the reduced number of input objects for the clustering method. Although the computation cost for each iteration of the K-means extension methods was not large, the same computation was required for each iteration; hence, a decrease in the computation cost for each iteration was an important factor for the rapid clustering. These experiments demonstrate that our method is useful for clustering very large data sets.

In Table 7, CF$^M$-based clustering methods for all real data sets were slower than non-summary-based clustering methods when this method used a memory size of 32 MB or more. This is because it identified a large number of microclusters. If the CF$^M$-based clustering method was performed based on a large memory size, it obtained a large number of microclusters, and then the overall clustering time was not reduced. However, our method reduced the clustering times in most cases because our method estimated a proper memory size by using an adaptive memory-size-increasing scheme for each data set.

AhmadK usually requires a large amount of execution time in most experiments because it makes an exceptionally large number of distance computations for all distinct value pairs of every attribute pair before carrying out clustering. On the other hand, our method required quite a small execution time, despite the fact that our method also computed the distances of all distinct value pairs for every attribute pair of the converted objects. Because the number of converted objects was fewer than the number of objects of the data sets,
The data sets for each summary scheme such as CF-M distance computations reduced the clustering time. The memory size of each data set used in the non-summary-based clustering methods is increased with an increase in the memory size. However, their purity, inverse purity, and Rand index were not enhanced as shown in Table 7. Thus, CF-M-based clustering methods are inefficient when a given memory is not suitable for summarizing the data sets.

Table 7 shows that our method clusters these data sets 2 to 10 times faster than CF-M-based clustering methods. This is because our adaptive memory-size-increasing scheme computed the proper memory size depending on the characteristics of each data set. In Table 8, the memory sizes of the MCF trees are much less than those of CF-M trees. This shows that MCF tree summarizes data sets much better than CF-M trees. As a result, the global clustering methods of our proposed scheme cluster the much less number of microclusters than those of CF-M-based clustering methods, which in turn significantly reduces the clustering times of our method.

**VI. CONCLUSIONS**

We proposed a novel summary-based clustering method that efficiently clusters very large mixed-type data sets by incorporating an adaptively increasing memory size for all data sets because it summarized the data set based on the given memory. The execution time of most CF-M-based clustering methods is increased with an increase in the memory size.
more accurate threshold estimation. A cluster of the mixed-type data sets is summarized in our proposed MCF vector, consisting of a CF vector and a histogram. Based on the MCF vector, we build the MCF tree using our proposed distance measure. Our adaptive memory-size-increasing scheme determines the proper initial memory size using a small portion of the objects for each data set (i.e., 0.06% - 3%). We demonstrated that our distance measure holds the triangle inequality property. It allows a range-query-based clustering method, ERC, to be used as our global clustering method. The scalability and flexibility of our clustering method were demonstrated using diverse large synthetic and real data sets. Consequently, our method clustered very large mixed-type data sets more rapidly than various existing clustering methods, while maintaining similar or better clustering accuracy.

REFERENCES

[1] A. A. Abbasi and M. F. Younis. A survey on clustering algorithms for wireless sensor networks. Comput. Commun., 30(14-15):2826–2841, 2007.
[2] C. C. Aggarwal. Data Mining - The Textbook. Springer, 2015.
[3] C. C. Aggarwal and C. K. Reddy. Educational and software resources for data clustering, 2013.
[4] C. C. Aggarwal and C. Zhai. A survey of text clustering algorithms. In C. C. Aggarwal and C. Zhai, editors, Mining Text Data, pages 77–128. Springer, 2012.
[5] A. Ahmad and L. Dey. A k-mean cluster algorithm for mixed numeric and categorical data. Data Knowl. Eng., 63(2):503–527, 2007.
[6] A. Ahmad and S. S. Khan. Survey of state-of-the-art mixed data clustering algorithms. IEEE Access, 7:31883–31902, 2019.
[7] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. Inf. Retr., 12(4):461–486, 2009.
[8] C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas. Randomized dimensionality reduction for k-means clustering. IEEE Trans. Inf. Theory, 61(2):1045–1062, 2015.
[9] J. Branke, M. Grobelnik, and M. Dladić. A survey of ontology evaluation techniques. In Proceedings of the conference on data mining and data warehouses (SKDD 2005), pages 166–170, 2005.
[10] R. Brinawy and N. Shiril. K-mixed prototypes: a clustering algorithm for relational data with mixed attribute types. In C. Hung and G. A. Papadopoulos, editors, Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8–12, 2019, pages 542–545. ACM, 2019.
[11] J. Chen and H. He. A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data. Inf. Sci., 345:271–393, 2016.
[12] J. Chen, X. Lin, H. Zheng, and X. Bao. A novel cluster center fast determination clustering algorithm. Appl. Soft Comput., 57:539–555, 2017.
[13] T. Chiu, D. Fang, J. Chen, Y. Wang, and C. Jeries. A robust and scalable clustering algorithm for mixed type attributes in large database environment. In D. Lee, M. Scholkinck, F. J. Provost, and R. Srikan, editors, Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26–29, 2001, pages 263–268. ACM, 2001.
[14] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu. Dimensionality reduction for k-means clustering and low rank approximation. In R. A. Servedio and R. Rubinfeld, editors, Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14–17, 2015, pages 163–172. ACM, 2015.
[15] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1):107–113, Jan. 2008.
[16] S. Ding, M. Du, T. Sun, X. Xu, and Y. Xue. An entropy-based density peaks clustering algorithm for mixed type data employing fuzzy neighborhood. Knowl. Based Syst., 133:294–313, 2017.
[17] M. A. B. Hajkacem, C. B. N’cir, and N. Essoussi. Mapreduce-based k-prototypes clustering method for big data. In 2015 IEEE International Conference on Data Science and Advanced Analytics, DSSA 2015, Campus des Cordeliers, Paris, France, October 19–21, 2015, pages 1–7. IEEE, 2015.
[18] M. A. B. Hajkacem, C. B. N’cir, and N. Essoussi. One-pass mapreduce-based clustering method for large mixed scale data. J. Intell. Inf. Syst., 52(3):619–636, 2019.
[19] Z. He, X. Xu, S. Deng, and B. Dong. K-histograms: An efficient clustering algorithm for categorical dataset. CoRR, abs/cs/0509033, 2005.
[20] C. Henning and T. F. Liao. How to find an appropriate clustering for mixed-type variables with application to socio-economic stratification. Journal of the Royal Statistical Society: Series C (Applied Statistics), 62(3):309–369, 2013.
[21] C. Hsu, C. Chen, and Y. Su. Hierarchical clustering of mixed data based on distance hierarchy. Inf. Sci., 177(20):4474–4492, 2007.
[22] C. Hsu and Y. Chen. Mining of mixed data with application to catalog marketing. Expert Syst. Appl., 32(1):12–23, 2007.
[23] C. Hsu and Y. Huang. Incremental clustering of mixed data based on distance hierarchy. Expert Syst. Appl., 35(3):1177–1185, 2008.
[24] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li. Automated variable weighting in k-means type clustering. IEEE Trans. Pattern Anal. Mach. Intell., 27(5):657–668, 2005.
[25] Z. Huang. Clustering large data sets with mixed numeric and categorical values. In Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD), pages 21–34. Singapore, 1997.
[26] A. A. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. ACM Comput. Surv., 31(3):264–323, 1999.
[27] J. Ji, T. Bai, C. Zhou, C. Ma, and Z. Wang. An improved k-prototypes clustering algorithm for mixed numeric and categorical data. Neurocomputing, 120:590–596, 2013.
[28] H. Jindi, M. A. B. Hajkacem, and N. Essoussi. Parallel k-prototypes clustering with high efficiency and accuracy. In M. Song, I. Song, G. Kotsis, A. M. Eoua, and I. Khalil, editors, Big Data Analytics and Knowledge Discovery - 22nd International Conference, DaWaK 2020, Bratislava, Slovakia, September 14–17, 2020, Proceedings, volume 12393 of Lecture Notes in Computer Science, pages 380–395. Springer, 2020.
[29] C. Li and G. Biswas. Unsupervised learning with mixed numeric and nominal data. IEEE Trans. Knowl. Data Eng., 14(4):673–690, 2002.
[30] C. Li, S. Jegelka, and S. Sra. Efficient sampling for k-determinantal point processes. CoRR, abs/1509.01618, 2015.
[31] J. Lim, J. Jun, S. H. Kim, and D. McLeod. A framework for clustering mixed attribute type datasets. In Proceedings of the 4th International Conference on Emerging Databases (EDEB 2012), Seoul, Korea, August 23-28, 2012. Citeseer, 2012.
[32] D. S. Modha and W. S. Spangler. Feature weighting in k-means clustering. Mach. Learn., 52(3):217–237, 2003.
[33] G. Philip and B. Ottaway. Mixed data cluster analysis: an illustration using cyriptic hooked-tang weapons. Archaeometry, 25(2):119–133, 1983.
[34] W. M. Rand. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association, 66(336):846–850, 1971.
[35] H. Ryu, S. Jung, and S. Pramanik. An improved clustering method over c++ tree using multiple range queries. IEEE Trans. Knowl. Data Eng., 32(9):1694–1706, 2020.
[36] R. S. Sangam and H. Om. Hybrid data labeling algorithm for clustering large mixed type data. J. Intell. Inf. Syst., 45(2):273–293, 2015.
[37] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In E. M. Nahum and D. Xu, editors, 2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud’10, Boston, MA, USA, June 22, 2010. USENIX Association, 2010.
[38] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In H. V. Jagadish and I. S. Mumick, editors, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4–6, 1996, pages 103–114. ACM Press, 1996.
[39] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. Data Min. Knowl. Discov., 1(2):141–182, 1997.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Citation information: DOI 10.1109/ACCESS.2021.3118413, IEEE Access
Hyeong-Cheol Ryu received his B.S. degree in computer science from Hanshin University, Korea in 2009. He received his M.S. and Ph.D. degrees in computer science and engineering from Sogang University, Korea in 2014 and 2020, respectively. Since 2020, he has been with SK hynix, Icheon, Korea, where he is currently a Senior Engineer in the NAND SE technology Group. His research interests include spatial databases and data mining.

Sungwon Jung received his B.S. degree in computer science from Sogang University, Seoul, Korea in 1988. He received his M.S. and Ph.D. degrees in computer science from Michigan State University, Michigan, USA in 1990 and 1995, respectively. He is currently a Professor in the Computer Science and Engineering Department at Sogang University. His research interests include mobile databases, spatial databases, data mining, and multimedia databases.