iVIBRATE: Interactive Visualization Based Framework for Clustering Large Datasets (Version 3)

Keke Chen Ling Liu
College of Computing, Georgia Institute of Technology
{kekechen, lingliu}@cc.gatech.edu

Abstract

With continued advances in communication network technology and sensing technology, there is an astounding growth in the amount of data produced and made available through the cyberspace. Efficient and high-quality clustering of large datasets continues to be one of the most important problems in large-scale data analysis. A commonly used methodology for cluster analysis on large datasets is the three-phase framework of "sampling/summarization − iterative cluster analysis − disk-labeling". There are three known problems with this framework, which demand effective solutions. The first problem is how to effectively define and validate irregularly shaped clusters, especially in large datasets. Automated algorithms and statistical methods are typically not effective in handling such particular clusters. The second problem is how to effectively label the entire data on disk (disk-labeling) without introducing additional errors, including the solutions for dealing with outliers, irregular clusters, and cluster boundary extension. The third problem is the lack of research about the issues for effectively integrating the three phases. In this paper, we describe iVIBRATE − an interactive-visualization based three-phase framework for clustering large datasets. The two main components of iVIBRATE are its VISTA visual cluster rendering subsystem, which invites human into the large-scale iterative clustering process through interactive visualization, and its Adaptive ClusterMap Labeling subsystem, which offers visualization-guided disk-labeling solutions that are effective in dealing with outliers, irregular clusters, and cluster boundary extension. Another important contribution of iVIBRATE development is the identification of special issues presented in integrating the two components and the sampling approach into a coherent framework, and the solutions to improve the reliability of the framework and to minimize the amount of errors generated throughout the cluster analysis process. We study the effectiveness of the iVIBRATE framework through a walkthrough example dataset of a million records and experimentally evaluate the iVIBRATE approach using both real-life datasets and synthetic datasets. Our results show that iVIBRATE can efficiently involve the user into the clustering process and generate high-quality clustering results for large datasets.

1 Introduction

Cluster analysis is a critical component in large-scale data analysis. Over the past decade, large datasets have been collected and analyzed in many application domains, varying from bioinformatics, information retrieval,

physics, geology, to marketing and business trend prediction. Many have reached the level of terabytes to petabytes [27, 20]. There is a growing demand for efficient and flexible clustering techniques that can adapt to the large datasets with complex cluster structure.

A dataset used in clustering is typically represented as a table D consisting of d dimensions (columns) and N records (rows). A record can represent an event, an observation or some meaningful entity in practice, while a dimension could be an attribute/aspect of the entity. Clustering algorithms try to partition the records into groups with some similarity measure [30]. A dataset can be large in terms of the number of dimensions (dimensionality), the number of records, or both. The problem of high dimensionality (hundreds or thousands of dimensions) is typically addressed by feature selection and dimensionality reduction techniques [12, 39, 59, 4]. In this paper, we will focus on cluster analysis for *numerical* datasets with a very large number of records ($>$ 1 million records) and a medium number of dimensions (usually $<$50 dimensions), assuming that the high dimensionality has been reduced before datasets entering the iVIBRATE framework for cluster analysis.

1.1 General Problems with Clustering Large Datasets

Several clustering algorithms have aimed at processing the *entire* dataset in linear or near linear time, such as WaveCluster [49], DBSCAN [16], and DENCLUE [24]. However, there are some drawbacks with these approaches.

(1)Time Complexity of Iterative Cluster Analysis. Typically, cluster analysis continues after the clustering algorithm finishes in a run, unless the user has evaluated, understood and accepted the clustering patterns or results. Therefore, the user needs to be really involved in the iterative process of "clustering and analysis/evaluation". In this process, multiple clustering algorithms, or multiple runs of the same algorithm with different parameter settings can be tested and evaluated. Even for a clustering algorithm with linear computational complexity, running such an algorithm on a very large dataset for multiple times could become intolerable. Moreover, most cluster validation methods have non-linear time complexity [23, 31, 15]. When performed on the entire large dataset, the validation of clusters hinders the performance improvement for the entire iterative process.

(2)Cluster Analysis on Representative Dataset vs. on Entire Dataset. Bearing the above problems in mind, a number of approaches were proposed to perform clustering algorithms on smaller sample datasets or data summaries instead of the entire large dataset. For example, CURE [21] applies random sampling to get the sample data and then runs a hierarchical clustering algorithm on the sample data. BIRCH [61] summarizes the entire dataset into a CF-tree and then runs a hierarchical clustering algorithm on the CF-tree to get clustering result. In fact, since the size of dataset is reduced with the sampling/summarization techniques, any typical clustering algorithms and cluster validation techniques that have acceptable non-linear computational complexity can be applied. This "sampling/summarization – iterative cluster analysis" framework has been commonly recognized as a practical solution to large-scale cluster analysis.

However, the above two-phase framework does not address the questions for the entire large dataset that are

frequently asked by some applications: 1) what is the cluster label for a particular data record which may not be in the representative dataset? 2) what are the data records in the entire dataset that belong to a particular cluster? Therefore, we also need to extend the intermediate clustering result to the entire dataset, which requires the third phase - labeling data on disk with the intermediate clustering result. Previous research on clustering with the three-phase framework has been primarily focused on the first two phases. Surprisingly, very few studies have considered the efficiency and quality of the disk-labeling phase.

Disk-labeling also provides the opportunity to review and correct the errors generated by the first two phases. For example, sampling/summarization tends to lose the small clusters in the representative dataset. When sampling is applied in the first phase, it is easy to understand that small clusters might be lost for small sample size. This is also true when summarization is done in a high granularity. When a CF-tree in BIRCH, for instance, is relatively small compared to the number of records, a leaf node will possibly cover a large spatial locality and we have to consider all small clusters in the leaf as one cluster. Although many applications only consider the primary clustering structure, i.e., the large clusters, small clusters may become significant for some applications. Thus, there is a need monitoring the small clusters possibly missed by the first phase.

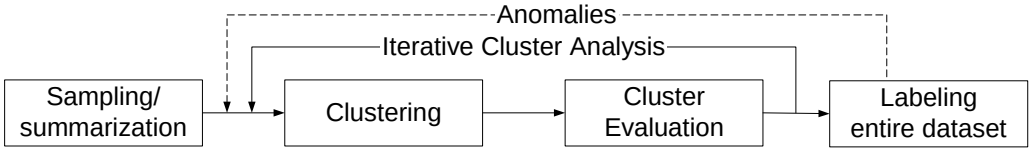

Figure 1: Three phases for cluster analysis of large datasets, (Sampling/summarization – Cluster Analysis – Disk Labeling)

(3) Problems with Irregularly Shaped Clusters. Many automated clustering algorithms work effectively in finding clusters in spherical or elongated shapes but they cannot handle arbitrarily shaped clusters well [21], neither can traditional validation methods, which are primarily statistical methods,effectively validate such clusters [23, 48].

Particularly, in some applications, irregularly shaped clusters may be formed by combining some regular clusters or by splitting one large cluster based on the domain knowledge. Most of the existing clustering algorithms do not allow the user to participate in the clustering process until the clustering algorithm is completed. Thus, it is inconvenient to incorporate the domain knowledge into the cluster analysis, or to allow the user to steer the clustering process that totally employs automated algorithms.

We observe that visualization techniques have played an important role in solving the problem of irregularly shaped clusters in large datasets. Some visualization-based algorithms, such as OPTICS [1], tried to find the arbitrarily shaped clusters, but they are often only applicable to small datasets and few studies have been performed for large datasets. The iVIBRATE approach described in this paper fills in this gap with the visualization-based three-phase framework and solves the particular problems related to the integration of the three phases under the framework.

(4)Problems with Disk Labeling. When disk labeling is used as the last phase of clustering large dataset,

it assigns a cluster label to each data record on disk based on the intermediate clustering result. Without an effective labeling phase, a large amount of errors can be generated in this process.

In general, the quality of disk-labeling depends on the precise description of cluster boundaries. All existing labeling algorithms are based on very rough cluster descriptions [31], such as a centroid or a set of representative cluster boundary points for a cluster. A typical labeling algorithm assigns each data record on disk to a cluster that has its centroid or its representative boundary points closest to this data record. Centroid-based labeling (CBL) uses the cluster center (centroid) only to represent a cluster; Representative-point-based labeling (RPBL) uses a set of representative points on cluster boundary to describe the cluster. The latter is better because it provides more information about the cluster boundary. With RPBL, the quality of boundary description mainly depends on the number of representative points, which could be very large for some irregular cluster shapes or large clusters. However, it is always not easy for the user to determine the sufficient number of representative points for a particular dataset.

In particular, the cluster boundary cannot be precisely defined with only the sample dataset. Cluster boundaries often continue to evolve as we incorporate more labeled records during the disk labeling phase. We name it the "cluster boundary extension" problem and will describe it in detail later.

1.2 The Scope and Contribution of the Paper

We have summarized four key problems in clustering large datasets: 1) the three-phase framework is necessary for reducing the time complexity of an iterative cluster analysis; 2) extending the clustering result on the representative dataset to the entire large dataset can raise significant problems; 3) clustering and validating irregularly shaped clusters in large datasets is important but difficult; and 4) existing disk-labeling algorithms may result in large errors primarily due to the imprecise cluster boundary description.

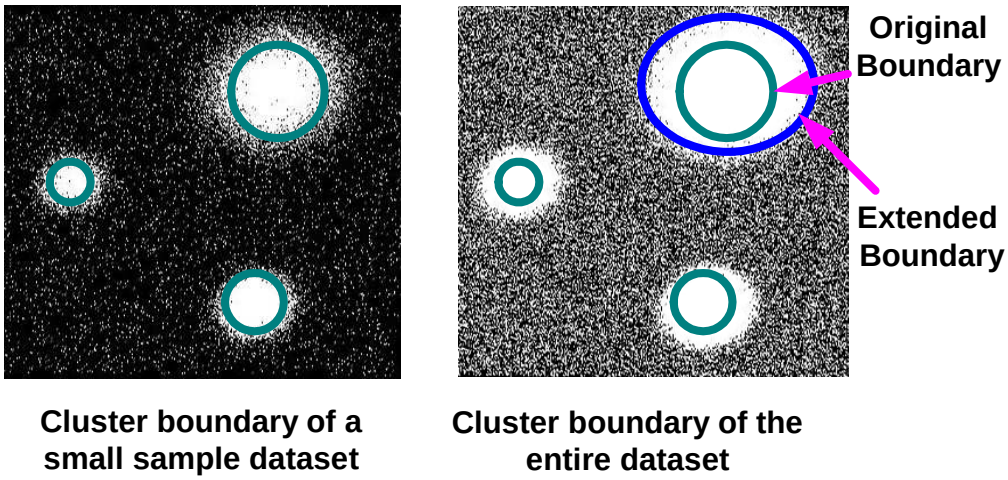

Figure 2: Comparing the cluster boundary of small and large dataset (data points are white)

We also explicitly identify that the problem of cluster boundary extension is a big challenge in the labeling

phase if sampling is applied in the first phase. Figure 2 shows the clusters evolving from the small ones in the representative dataset to the larger ones in the entire dataset, where boundary extension results in significant difference in cluster definition. The point density over the initial boundary could increase significantly as the number of labeled records increases, which naturally leads to the boundary extension problem. Especially, when the sample size is much smaller than the size of the large dataset, e.g. $< 1\%$ of the original data records, boundary extension can cause significant errors if labeling algorithms fail to adapt to it.

Boundary extension can also cause two additional problems. 1) For the regular spherical clusters as shown in Figure 2, existing labeling algorithms usually either assign all outliers to the nearby clusters, or treat the cluster members in the extended areas as outliers. As a result, none of them can deal with outliers and boundary extension effectively. For irregular cluster boundary, the situation can be worse. 2) Boundary extension might also result in the overlapping of different clusters, which are originally separated in the representative set. Monitoring boundary extension allows us to recheck and adjust the initial cluster definition.

To address all of the above problems, we propose the iVIBRATE framework – an interactive visualization based three-phase framework for clustering large datasets. The iVIBRATE framework includes the three phases "Sampling – Visual Cluster Analysis – Visualization-based Adaptive Disk-labeling". In this paper, we introduce the two main components: visual cluster analysis and visualization-based adaptive disk-labeling, while focusing on the important issues in integrating the three components in the iVIBRATE framework. We also demonstrate how to analyze very large datasets with the iVIBRATE framework.

In the clustering phase, we use visual cluster analysis, including visual clustering and visual cluster validation, to allow the user to participate in the iterative cluster analysis, reducing both the length of single iteration and the number of iterations. We develop a visual cluster rendering system, VISTA, to perform "visual cluster analysis". The VISTA system interactively visualizes multi-dimensional datasets (usually $<$50 dimensions). It allows the user to interactively observe potential clusters in a series of continuously changing visualizations. More importantly, it can incorporate the algorithmic clustering results, and serve as an effective validation and refinement tool for irregularly shaped clusters.

In the disk-labeling phase, we develop the Adaptive ClusterMap Labeling subsystem. ClusterMap encodes the irregular cluster boundaries defined on the visualization that is generated by the VISTA subsystem. The algorithm automatically adapts to the boundary extension phenomenon, clearly distinguishes outliers, continuously detecting irregular shaped clusters, and visually monitoring the anomalies in labeling process. As a result, the Adaptive ClusterMap labeling phase generates fewer errors than the existing disk-labeling algorithms.

When the three phases are integrated in the iVIBRATE framework, errors caused by the improper operations in the prior phases may propagate to the later phases. Visualization helps to detect and reduce such errors. We identify and analyze the issues related to the integration of the phases, and develop the theory and tools to monitor the possible errors. The iVIBRATE framework is evaluated with real and synthetic datasets and the experimental results show that iVIBRATE can take advantage of visualization and user interaction to generate high-quality clustering results for large datasets.

The rest of the paper is organized as follows. Section 2 reviews some related work. Section 3 presents the

design principles and components of iVIBRATE framework. Section 4 and Section 5 briefly introduce the two main components of the framework: the VISTA visual cluster rendering system and the ClusterMap labeling algorithms. Section 6 describes the problems and the solutions in the integration of the three phases. Section 7 reports some experimental results, demonstrating that the iVIBRATE approach can not only effectively discover and validate irregular clusters, but also effectively extend the intermediate clustering result to the entire large dataset. We also present an detailed example, showing how to explore a very large real dataset: census dataset with the iVIBRATE framework, in section 8.

2 Related Work

Clustering Large Data. We have described four challenges related to clustering large datasets: time complexity (scalability), sampling/summarization based clustering, irregular clusters and disk-labeling. Although each of these issues has been studied, there is surprisingly little study on how they impact on the cluster quality when sampling/summarization, iterative cluster analysis, and disk labeling are integrated into a unifying framework for exploring the complex cluster structures in large datasets.

Concretely, time complexity of clustering algorithm has been addressed from early on. K-means algorithm [30] is the most popular algorithm with linear time complexity. Most studies related to K-means assume that the clusters are in spherical shapes. Recently, there are some other algorithms [49, 16, 24] having started looking at the problem of clustering irregular clusters in linear/near-linear time.

Dealing with arbitrarily shaped clusters is well-recognized as a challenging problem in clustering research community. A number of clustering algorithms have aimed at this particular problem, such as CURE [21], CHAMELEON [33], DBSCAN [16], DBCLASD [56], WaveCluster [49], DENCLUE [24] and so on. The semi-automatic algorithm OPTICS [1], derived from the DBSCAN algorithm, shows that visualization can be very helpful in cluster analysis. However, all these algorithms are known to be effective only in low dimensional (typically, $<$10D) datasets or in small/medium datasets with thousands of records.

A general cluster analysis framework is described in a review paper [31], showing that cluster analysis is usually an iterative process. One approach to address the scalability of iterative clustering analysis is the use of the "sampling(summarization) – clustering–labeling" framework, represented by CURE [21] and BIRCH [61]. However, the labeling phase and interactions between the phases have not been sufficiently addressed.

As far as the summarization/sampling phase is concerned, instead of using BIRCH summarization, Bradley et al. [5] suggest using sufficient statistics to model the data summary. In comparison to summarization, sampling is used more extensively in data analysis: commercial vendors of statistical packages (e.g., SAS) typically use uniform sampling to handle large datasets. Vitter's reservoir sampling [53] represents an efficient uniform sampling technique. The main problem with uniform sampling is the loss of small clusters. CURE proposed a method to estimate the minimum sample size if the size of entire large dataset and the smallest size of clusters are known. However, the minimum sample size shall increase dramatically with the increase of dataset size. Thus, for a fixed sample size, it is also necessary to develop methods monitoring the small clusters in order to

maintain the consistency in cluster analysis.

Another popular sampling approach is density-biased sampling proposed by Palmer et al. [42]. A density-biased sampling preserves small clusters in the sampling process. However, this technique also skews the actual size of large clusters, introducing too much inconsistency between the clusters in the sample set and the actual clusters in the entire large dataset. It raises particular problems in the later two phases, which are not our focus in this paper.

The existing disk labeling solutions heavily depend on the concrete cluster representations generated at the iterative cluster analysis phase. Existing cluster representations can be classified into four categories [31] : centroid-based, boundary-point-based (representative-point-based), classification-tree-based and rule-based representations. The centroid representation is the most popular one. Many clustering algorithms in fact only generate cluster centroids, for example, K-means and most hierarchical algorithms. For boundary-point-based representations, good representative boundary points are often difficult to extract. The most typical algorithm for generating the representative points is CURE [21]. Classification-tree-based and rule-based representations are equivalent (each path in the classification tree can be represented as a rule), however, both are inconvenient to describe high-dimensional data or complicated clustering structure.

Document Clustering in IR and Linkage-based clustering in Network Analysis. Most of the research on clustering large datasets can be roughly categorized into three areas: scientific/business data clustering [30, 31], document clustering [55, 13, 45, 29, 46, 60, 51], and linkage-based clustering for large scale network analysis [31, 22, 41].

In scientific/business data clustering, the data is already formalized as a set of multi-dimensional vectors (i.e., a table). However, in document clustering, the original data is text data. Most document clustering techniques are focused on the two steps before applying clustering algorithms: extracting keywords/constructing the numerical features [59], and defining a suitable similarity measure [3]. Given the vector representation of documents and the similarity measure, document clustering is to some extent similar to scientific/business data clustering. Since document collections have become larger and larger with the wide spread of Internet-based applications, we expect that the iVIBRATE framework described in this paper can also be extended to clustering large sets of documents.

Graph mining or linkage-based clustering [31] has received a growing interest in the recent years due to increased interests in analyzing large-scale networks, such as peer to peer online communities [43], and social networks [41]. Linkage-based clustering is also used in clustering categorical datasets [22]. Most of the business/scientific data clustering algorithms utilize the distances between multi-dimensional data points (records) to compute and derive data clusters, while most of the linkage-based clustering algorithms utilize the node connectivity as a main measurement to understand and derive the interesting clustering structure of the network. Thus, linkage-based clustering algorithms aim at finding communities in networks − groups of vertices within which connections are dense but between which connections are sparser.

A commonalty of data clustering, document clustering, and node clustering is the fact that they all emphasize on efficient algorithms to speed up the clustering process of large datasets. However, the subtle differences

between distance based measure and connectivity-based measure may influence how the clustering algorithms are devised and what factors are critical to the performance of the algorithms. Due to the scope of our paper, we will confine our discussion to the general clustering problem to the business and scientific datasets.

Visualization of Multidimensional Data. Information visualization has demonstrated great advantages in multi-dimensional data analysis. Here we only discuss the scatter-plot-based techniques because they are the most intuitive techniques for cluster visualization. The early research on projection-based general data visualization is the Grand Tour [2]. The Grand Tour tries to find a series of smoothly changed projections that map data to two orthogonal dimensions, so that the user can look at the high-dimensional data from different perspectives. In order to reduce the huge search space for cluster visualization, Projection Pursuit is also used to identify some of interesting projections [11]. Yang [58] utilizes the Grand Tour technique to show projections in an animation. Dhillon, et al. [14] aimed at precise visualization of the clusters, but the technique is only effective for 3 clusters. When more than 3 clusters exist, it requires the help of the Grand Tour technique. The above systems aim at visualizing the datasets in continuously changed projections, which is similar to our VISTA system. However, it is well known that generating continuously changing visualizations in Grand Tour systems often involves complicated computation, and their visualization models are generally not intuitive to users. Most importantly, they do not fully utilize the power of interaction and heuristic rendering rules. Compared to the Grand Tour models, the VISTA visualization model has several advantages: 1) it provides convenient parameter adjustments; 2) it is enhanced by the heuristic rendering rules and the intuitive interpretation about the rules for finding the satisfactory cluster visualization; 3) continuously changing the VISTA visualization is very easy and fast, either in ARR mode or ADDR mode (see section 4.3), which produces the effect of animation at low cost.

Different from the dynamic visualization systems, like the Grand Tour and VISTA, there are static multidimensional visualization techniques such as Scatterplot Matrices, coplots, Parallel Coordinates [28], Glyphs [38], multidimensional stacking [36], prosection [9] and FastMap based visualization [17]. A nice tool, XvmdvTool [54], implements some of the above static visualization techniques. Some techniques are extended to specifically visualize the clustering structures discovered by clustering algorithms, such as IHD [57] and Hierarchical Clustering Explore [47]. However, these techniques are not specifically designed to address the difficult clustering problems: irregularly shaped clusters, domain-specific clustering structure, and problems in labeling clusters in very large datasets. Most of them are also limited by the dimensionality (10-20 dimensions at maximum).

Star Coordinates [32] is a visualization system designed to interactively visualize and analyze clusters. We utilize the form of Star Coordinates and build the α-mapping model in our system. α-mapping model extends the ability of the original mapping used in Star Coordinates, and the mechanism of visual rendering behind this model can be systematically analyzed and understood [7]. RadViz [26] utilizes the same coordinates system with a non-linear mapping function, which makes it difficult to interpret the visual clusters in the generated visualization. HD-Eye [25] is another interesting interactive visual clustering system. It visualizes the density-plot of the interesting projection of any two of the k dimensions. It uses icons to represent the clusters and the relationship between the clusters. However, it is difficult for the user to synthesize all of the interesting

2D projections to find the general pattern of the clusters. In fact, visually determining the cluster distribution solely through user interaction is not necessary. A more practical approach is to incorporate all available clustering information, such as algorithmic clustering results and the domain knowledge, into the visual cluster exploration.

3 The iVIBRATE Framework

In this section, we first briefly give the motivation and the design ideas of the iVIBRATE development, and then describe the components and working mechanism of iVIBRATE.

Motivation. In the three-phase framework, cluster analysis involves the ”clustering - analysis/evaluation” iteration, which can be concretely described in the following steps:

1. Run a clustering algorithm with the initial setting of parameters.
2. Analyze the clustering result with statistical measures and the domain knowledge.
3. If the result is not satisfactory, adjust the parameters and re-run the clustering algorithm, or use another algorithm, then go to Step 2 to evaluate the clustering result again until the satisfactory result is obtained.
4. If the result is satisfactory, then perform post-processing, which may include labeling the data records on disk.

Open problems. We first discuss a number of open problems in the above steps, and then we describe how iVIBRATE addresses these problems. Traditional statistical methods, such as variance and intra/inter cluster similarity, are typically used in Step 2, which assume that the shape of cluster structure is hyper-sphere or hyper-ellipse. As a result, these traditional statistical methods have difficulty in effectively validating the irregular cluster shapes [15, 23]. Moreover, with automated algorithms, it is almost impossible to incorporate the domain knowledge. The critical task in step 3 is to learn and determine the appropriate parameter settings. For example, CURE [21] requires the number of representative points and the shrink factor. DBSCAN [16] needs proper ε and *MinPts* to get satisfactory clusters. DENCLUE [24] needs to define the smoothness level and the significance level. These parameter settings are different from dataset to dataset and depend primarily on the user to try different parameters and find the “best” set of parameters by hand. Therefore, there is a need to help the user to easily find the appropriate parameter setting, when automated algorithms are applied. Finally, a coarse labeling algorithm tends to deteriorate the intermediate clustering result as we have discussed.

Bearing these problems in mind, we observed that, if the step 2 and 3 can be carefully combined together, which means that the user can perform evaluation in the course of clustering and be able to refine the clusters at the same time, the length of an iteration would also be greatly reduced. In addition, the user would understand more about the dataset and thus be more confident in their judgment of the clustering results. This motivates us to develop and promote the interactive visual cluster rendering approach.

Cluster Visualization and Visual Validation. Cluster visualization can improve the understanding of the clustering structure. Former studies [35] in the area of visual data exploration support the notion that visual exploration can help in cognition. Visual representations can be very powerful in revealing trends, highlighting outliers, showing clusters, and exposing gaps. According to the paper [50], with the right coding, human pre-attentive perceptual skills enable users to recognize patterns, spot outliers, identify gaps and find clusters in a few hundred milliseconds. In addition, it does not require the knowledge of complex mathematical/statistical algorithms or parameters [34].

Visualization is known to be the most intuitive method for validating clusters, especially clusters in irregular shape. Since the geometry and density features of clusters, which are derived from the distance (similarity) relationship, determine the validity of the clustering results, many clustering algorithms in literature use 2D-plot of clustering results to visually validate their effectiveness on 2D experimental datasets. However, visualizing high-dimensional datasets keeps as a challenging problem.

Static vs. Dynamic Data Visualization. In general, multi-dimensional data visualization can be categorized into static visualization or dynamic visualization. Static visualization displays the data with a fixed set of parameters, while dynamic cluster visualization allows the user to adjust a set of parameters, resulting in a series of continuously changed visualizations. It is commonly believed that static visualization is not sufficient in visualizing clusters [34, 50], and it has been shown that clusters can hardly be satisfactorily preserved in a static visualization [11, 14]. Therefore, we consider using interactive dynamic cluster visualization in the iVIBRATE framework. We observe that a cluster can always be preserved as a point-cloud in visual space through linear mappings. The only problem is that, these point-clouds may overlap with one another and to find certain mapping that can satisfactorily separate the point clouds is mathematically complex. In the iVIBRATE framework, we incorporate the combination of visual cluster clues and interactive rendering into the iterative clustering analysis, and refine algorithmic clustering results with heuristic rendering rules, which enables us to identify these point-cloud overlaps quickly and intuitively.

Visualization-based Disk-labeling. Another unique characteristic of iVIBRATE is its visualization-based disk-labeling algorithm. We argue that a fine cluster visualization of a dataset can serve as the visual clustering pattern of this dataset, where the cluster boundary can be precisely described and most outliers can be clearly distinguished. We develop the basic ClusterMap labeling algorithm for obtaining better description of cluster boundary and higher quality of disk-labeling. In order to solve the problem of cluster boundary extension, we also extend the basic ClusterMap algorithm to the Adaptive ClusterMap labeling algorithm.

Components and Working Mechanism. Figure 3 sketches the main components of iVIBRATE framework. We briefly describe each of the main components and the general steps used to analyze the clusters in large datasets.

- **Visual Cluster Rendering** The VISTA system can be used independently to render the clusters in a dataset without incorporating any external information. It can also visualize the result of an automated clustering algorithm or use the result to guide the interactive rendering. By interactively adjusting the parameters, the user can visually validate the algorithmic clustering results through continuously changing

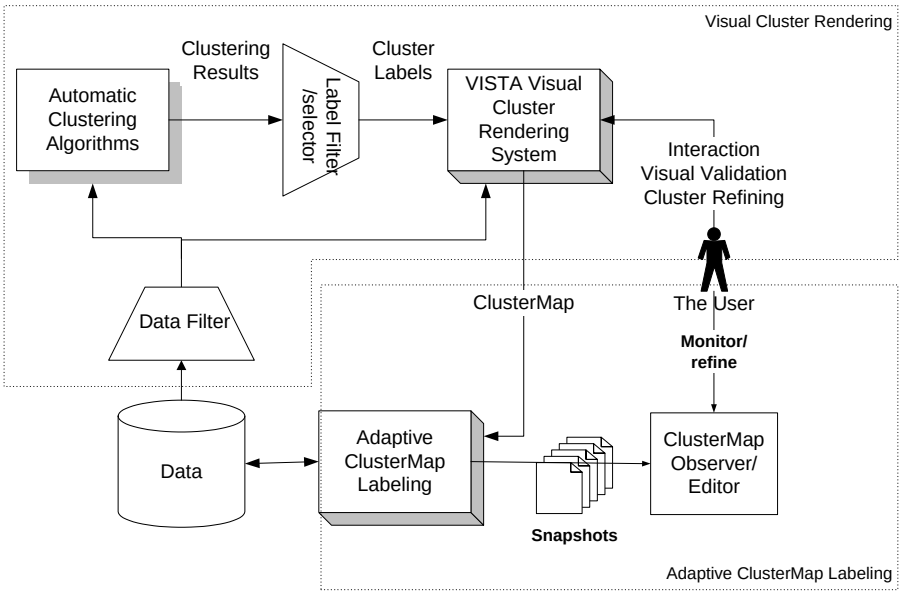

Figure 3: The iVIBRATE framework

visualizations. Using a couple of rendering rules, which are easy to learn, the user can quickly identify cluster overlaps and improve the cluster visualization as well. In addition, it also allows the user to conveniently incorporate domain knowledge into cluster definition through visual rendering operations. Semi-automated rendering method is also provided for larger number of dimensions ($> 50D$ and $< 100D$).

Data Filter prepares the data for visualization. It handles missing values and normalizes the data. If the dimensionality is too high, dimensionality reduction techniques or feature selection might be applied to get a manageable number of dimensions. When the size of dataset grows past a million items that cannot be easily visualized on a computer display, Data Filter also uniformly samples the data to create a manageable representative dataset. It also extracts certain relevant subsets, for example, one specific cluster, for detailed exploration.

Label Selector selects the clustering result that will be used in visualization as clustering clues or for validation. While a clustering algorithm finishes, it assigns a label to each data record. Label Selector extracts part of the labels corresponding to the data records extracted by Data Filter.

- **Adaptive ClusterMap Labeling** In the iVIBRATE framework, we introduce ClusterMap $-$ a new cluster presentation, and the associated disk-labeling methods. ClusterMap makes the labeling result highly consistent with the intermediate cluster distribution. Adaptive ClusterMap labeling algorithm then automatically adjusts the cluster boundary according to the accumulation of labeled data records at the boundary areas.

  ClusterMap Observer is an interactive monitoring tool. It observes the snapshots, i.e. the changing ClusterMaps during labeling. The snapshots may provide information about the bias of the representative sample set, for example, the missing small clusters, and the anomalies in labeling as shown in section

6.2.

A user of iVIBRATE will perform the cluster analysis in the following seven steps. 1) The large dataset is sampled to get a subset in a manageable size (e.g., thousands or tens of thousands records). 2) The sample set is used as an input to the selected automatic clustering algorithms and to the VISTA visual rendering subsystem. The algorithmic clustering result provides helpful information in the visual cluster rendering process. 3) The user interacts with the VISTA visual cluster rendering system to find the satisfactory visualization, which visualizes the clusters in well-separated areas. Since human vision is very sensitive to the gap between point clouds, which implies the actual boundary of clusters, the interactive rendering works very well in refining vague boundaries or irregular shaped clusters. 4) A ClusterMap is then defined on the satisfactory cluster visualization and used as the initial pattern in ClusterMap labeling. 5) The labeling process will adapt the boundary extension and refine the cluster definition in one pass through the entire dataset. An additional pass might be needed to reorganize the entire dataset for fast processing of queries. During the labeling process, snapshots are saved periodically, which are then used to monitor the anomalies during the labeling process. 6) The user can use the ClusterMap Observer to check the snapshots and refine the extended ClusterMap. 7) To further observe the small clusters that may be omitted in the sampling process, the data filtering component is used to filter out the labeled outliers and performs sampling/visual rendering on the sampled outliers again (for details, see section 6.3).

In the following sections, we will introduce the two subsystems, with a focus on the integration of the main components into the framework.

4 VISTA Visual Cluster Rendering System

A main challenge in cluster visualization is cluster preservation, i.e., visualizing multi-dimensional datasets in 2D/3D visual space, while preserving the clustering structure. Previous studies have shown that preserving cluster structure precisely in static visualization, if not impossible, is very difficult and computationally expensive [34, 11, 58, 14, 50]. An emerging practical mechanism to address this problem is to allow the user to interactively explore the dataset [34] and to distinguish the visibly inaccurate cluster structure, such as cluster overlapping, broken clusters and false clusters (the situation where the outliers in the original space are mapped to the same visual area and thus form a false visual cluster) through visual interactive operations.

The iVIBRATE visual cluster rendering subsystem (VISTA) is designed to be a dynamic visual cluster exploration system. It uses a visualization model, characterized by the max-min normalization and the α-mapping to produce a linear transformation that maps each multi-dimensional data point onto a data point in 2D visual space. This mapping model provides a set of visually adjustable parameters, such as the α parameters. By continuously changing one of the parameters, the user can see the dataset from different perspectives. Since the linear mapping does not break the clusters, the clusters in multi-dimensional space are still visualized as dense point clouds (the "visual clusters") in 2D space. And the visible "gaps" between the visual clusters in 2D visual space indicate the *real gaps* between point clouds in the original high dimensional space. However,

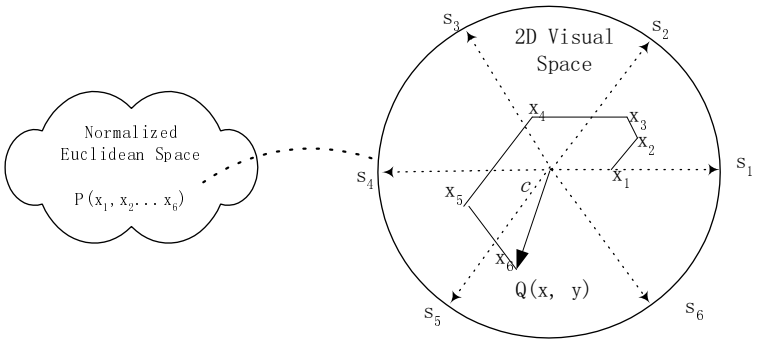

Figure 4: Illustration of α-mapping with $k = 6$

overlaps between the visual clusters in the 2D space, i.e. the point clouds, may occur with certain parameter settings. We have developed a set of interactive operations and designed several heuristic rendering rules in order to efficiently distinguish the visual cluster overlaps. These developments have shown to be quite effective in achieving desired rendering efficiency.

Since Euclidean distance is the most commonly used distance measure in applications, the current prototype of VISTA subsystem supports clustering with Euclidean distance. For the convenience of presentation, in the rest of the paper Euclidean distance is used as the default similarity measure. Datasets with other distance measures can be approximately transformed to Euclidean datasets with techniques like multidimensional scaling [12], which will be a part of VISTA extensions in the future work.

4.1 The Visualization Model

The VISTA visualization model consists of two linear mappings – max-min normalization followed by α-mapping. For better understanding of the iVIBRATE framework, we briefly introduce the two as follows. Interested readers can refer to the paper [7] for details.

Max-min normalization is used to normalize the columns in the datasets in order to eliminate the dominating effect of large-valued columns. For a column with value bounds [min, max], max-min normalization scales a value v in the column into [-1, 1] as follows:

$$v' = \frac{2(v - \min)}{\max - \min} - 1 \tag{1}$$

where v is the original value and v' is the normalized value. **α-mapping** maps k-D points onto the 2D visual space with the convenience of visual parameter tuning. We describe α-mapping as follows. Let a 2D point $Q(x, y)$ represent the image of a k-dimensional (k-D) max-min normalized data point $P(x_1, \ldots x_i, \ldots, x_k)$, $x_i \in [-1, 1]$ in 2D space. $Q(x, y)$ is determined by the average of the vector sum of the k vectors $\vec{s}_i\dot{x}_i$, where $\vec{s}_i = (\cos(\theta_i), \sin(\theta_i)), i = 1 \ldots k$ and $\theta_i \in [0, 2\pi]$ are the star coordinates [32] that represent the k dimensions

in 2D visual space. Formula 2 defines α-mapping.

$$A_{(\theta_1,\ldots,\theta_k)}(x_1,\ldots,x_k,\alpha_1,\ldots,\alpha_k) = (c/k)\sum_{i=1}^{k}\alpha_i x_i \vec{s_i} - \vec{o} \tag{2}$$

i.e. a 2D point $Q(x,y)$ is determined by

$$\{x,y\} = \{(c/k)\sum_{i=1}^{k}\alpha_i x_i \cos(\theta_i) - x_0, (c/k)\sum_{i=1}^{k}\alpha_i x_i \sin(\theta_i) - y_0\} \tag{3}$$

Here, $\alpha_i (i = 1\ldots k, \alpha_i \in [-1,1])$ provides the visually adjustable parameters, one for each of the k dimensions. $\alpha_i \in [-1,1]$ covers a considerable range of mapping functions. Experimental results show that this range combined with the scaling factor c is effective enough for finding satisfactory visualization. θ_i is set to $2i\pi/k$ initially and can be adjusted afterwards. We also proved that adjusting θ values is often equivalent to a pair of α adjustment plus zooming [7]. Thus, it is not necessary to change θ_i in practice. $\vec{o} = (x_0, y_0)$ is the center of the display area.

α-mapping is a linear mapping, with any fixed set of α values. Without loss of generality, we set the center translation (x_0, y_0) as $(0,0)$. The mapping $A_{\alpha_1=a_1,\ldots,\alpha_k=a_k}(x_1,\ldots,x_k)$ can be represented as the following transformation.

$$A_{\alpha_1=a_1,\ldots,\alpha_k=a_k}(x_1,\ldots,x_k) = \begin{bmatrix} \cos(\theta_1)\cdots\cos(\theta_k) \\ \sin(\theta_1)\cdots\sin(\theta_k) \end{bmatrix} \begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_k \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix}$$

It is known that linear mapping does not break clusters but may cause cluster overlaps [34, 19]. Since α-mapping is linear, there are no "broken clusters" in the visualization, i.e., the visual gaps between the point clouds reflect the real gaps between the clusters in the original high-dimensional space. All we need to do is to separate the possibly overlapped clusters, which can be achieved with the help of dynamic visualization through interactive operations.

The mapping is adjustable by α_i. By tuning α_i continuously, we can see the influence of the i-th dimension to the cluster distribution through a series of smoothly changing visualizations, which usually provides important clustering clues. The dimensions that are important to clustering will cause *significant changes* to the visualization as the corresponding α values are continuously changed.

α-mapping based visualization is implemented in the VISTA subsystem as shown in Figure 5. The coordinates are arranged around the display center and the α-widgets are designed for interactively adjusting each α value. However, the above visual design also limits the number of dimensions that can be visualized and manually manipulated. In the current prototype of VISTA, users can comfortably manually render up to 50 dimensions. Although the system can visualize more than 50 dimensions, we suggest using the semi-automated rendering method instead that will be introduced in section 4.3.

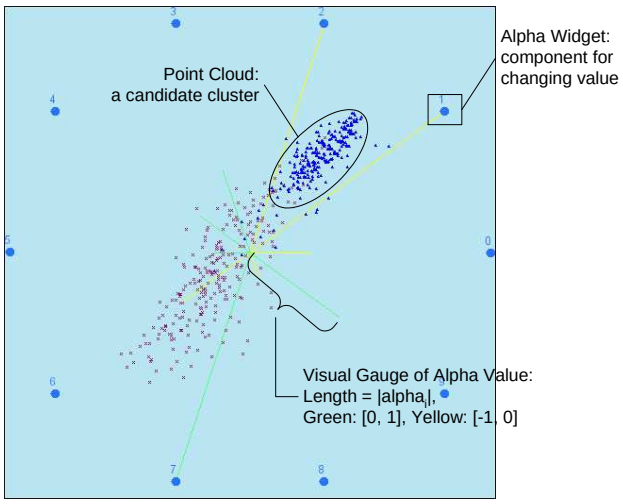

Figure 5: An implementation of α-mapping

Comparison with RadViz visualization model It is worth mentioning that the RadViz system [26] is also based on star coordinates, however, it uses a totally different mapping model. Let $P(x_1, \ldots x_i, \ldots, x_k)$ be the normalized point as above.

$$RV(x_1, \ldots, x_k) = \frac{\sum_{i=1}^{k} x_i \vec{s_i}}{\sum_{i=1}^{k} x_i} - \vec{o} \tag{4}$$

From Equation 4, we can see that RadViz mapping normalizes the contribution of each x_i by all dimensional values. The factor $(\sum_{i=1}^{k} x_i)^{-1}$ renders the mapping as a *non-linear* mapping, which leaves the "visual clusters" difficult to interpret. In fact, the original RadViz visualization is also static – as long as the ordering of k dimensions is determined a unique visualization is generated. Although it is easy to add a set of similar "α" parameters into the model, it depends on further study to develop or understand the potential interactive rendering rules. Since the two mapping models are totally different, our rendering rules and methods presented in the next sections and the paper [7] cannot be easily applied to the RadViz mapping model.

4.2 The Rules for Interactive Visual Rendering

To understand the basic visual rendering rules, we should investigate the dynamic properties of the visualization model, especially, the most important interactive operation – α-parameter adjustment (or simply, α-adjustment). α-adjustment changes the parameters defined in Eq. (2). Each change refreshes the visualization in real time (about a couple of hundred milliseconds, depending on different hardware configurations and the size of dataset), generating dynamically changing visualizations. α-adjustment enables the user to find the *dominating dimensions*, to observe the dataset from different perspectives, and to distinguish the real clusters from cluster overlaps in continuously changing visualizations.

Continuous α-parameter adjustment of one dimension reveals the effect of this dimension on the entire visualization. Let $X(x_1, \ldots, x_k)$ and $Y(y_1, \ldots, y_k)$, $x_i, y_i \in [-1, 1]$ represent any two normalized points in k-D

space. Let $\|\vec{v}\|$ represent the length of vector $\vec{v}$. We define the *visual distance* between X and Y is:

$$\begin{aligned} vdist(X,Y) &= \|A(x_1,\ldots,x_k,\alpha_1,\ldots,\alpha_i,\ldots,\alpha_k) - A(y_1,\ldots,y_k,\alpha_1,\ldots,\alpha_i,\ldots,\alpha_k)\| \\ &= \|(c/k)\sum_{i=1}^{k}\alpha_i(x_i-y_i)\vec{s_i}\| \end{aligned} \quad (5)$$

which means if x_i and y_i are close, changing α_i does not change the visual distance between X and Y a lot – the dynamic visual effect is that X and Y are moving together when α_i changes. Meanwhile, neighboring points in k-D space also have similar values in each dimension as Euclidean distance is employed. Thus, we can conclude that the neighboring points in k-D space, which should belong to one cluster, not only are close to each other in 2D space, but also tend to move together in any α-adjustment; while those points that are far away from each other in k-D space may move together in some α-adjustment but definitely not in all α-adjustments. This property makes α-adjustment very effective in revealing the visual cluster overlaps. In addition, point movement can also reveal the value distribution of individual dimension. If we adjust the α value of the dimension i only, the point movement can be represented by:

$$\begin{aligned} \Delta(i) &= A(x_1,\ldots,x_k,\alpha_1,\ldots,\alpha_i,\ldots,\alpha_k) - A(x_1,\ldots,x_k,\alpha_1,\ldots,\alpha_i',\ldots,\alpha_k) \\ &= (c/k)(\alpha_i-\alpha_i')x_i\vec{s_i} \end{aligned} \quad (6)$$

which means that the points having larger x_i will be moving "faster" along the i-th coordinate, and those having the similar x_i moving in a similar way. The initial setting of α values may not reveal the distribution of an individual dimension as Figure 6 shows. However, by looking at the density centers (the moving point cloud) along the i-th axis as α_i changes, we can easily estimate the value distribution along i-th dimension. In Figure 6, we sketch that point movement and point distribution can be interpreted intuitively with each other.

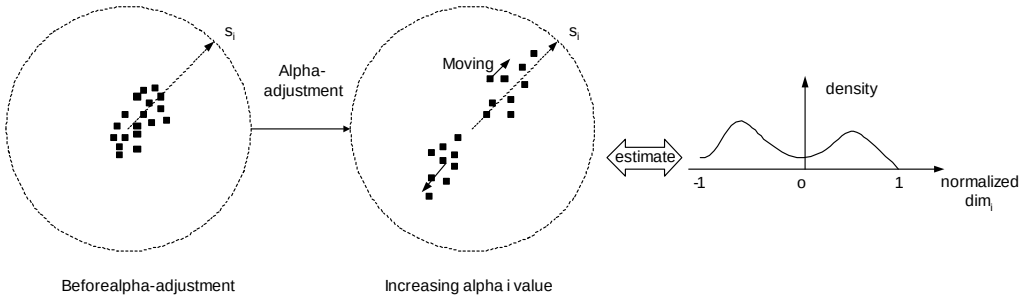

Figure 6: α-adjustment, dimensional data distribution and point movement

In interactive visual rendering, some dimensions show "significant change" on visualization in continuous α-adjustment, i.e., changing its α value results in distinct point clouds moving in different directions, or causes the visible "gaps" between point clouds to emerge. These dimensions play important roles in visual cluster rendering and thus we name them as "visually dominating dimensions", and the others as "the fine-tuning dimensions". The dominating dimensions usually have skewed distributions, where more than one distinctive mode exist on the distribution curve. For example, dimensions with near uniform distribution are definitely not

dominating dimensions and dimensions with normal distribution are also less likely to be dominating, however, possibly useful in refinement of visualization.

Since the main goal of VISTA interactions is to distinguish the possible visual cluster overlaps, we can apply the following rules in visual rendering:

Visual Rendering Rule 1. *Sequentially render each dimension. If the dimension is a visually dominating dimension, increase its α value to certain degree so that the main point clouds are satisfactorily distinguished.*

Visual Rendering Rule 2. *Use the fine-tuning dimensions to polish the visualization. Adjust their α values finely so that the visualization clearly shows the cluster boundaries.*

Guided by the above simple visual rendering rules, a trained user can easily find the satisfactory visualizations. While combined with the cluster labels that are generated by automatic clustering algorithms (for example, K-Means algorithm), the rendering becomes even easier. During the rendering process, we can intuitively validate the algorithmic clustering results and conveniently incorporate the domain knowledge into the clustering process [7], which are difficult for most automated clustering algorithms.

4.3 Semi-automated Rendering

When the number of dimensions grows to a considerably large number (> 50 and < 100 dimensions), manually rendering the dimensions becomes a difficult job. In the VISTA subsystem, we provide a semi-automated rendering method to automate the rendering of this type of datasets. Together with the visual rendering rules we have presented, the semi-automated rendering method can be quite efficient.

Concretely, our semi-automated rendering is performed in two stages: automatic random rendering (ARR) followed by automatic dimension-by-dimension rendering (ADDR). A simple version of random rendering is defined as follows. Let a dataset S have d dimensions and N records. Each dimension i ($1 \leq i \leq k$) is associated with an initial α value, say α_i. Random rendering can be done in any number of rounds until some rough pattern of cluster distribution is observed. In each round, the α_i value is changed by a small constant amount ϵ, ($0 < \epsilon < 1$), but the direction (increase or decrease) is randomly chosen for each α_i. Since the α values are bounded by 1 and -1, the change is "bounced" back at the ends. By changing α values in this way, rather than randomly assigning them in each round, we can observe that the visualization is more smoothly changed. This type of continuity between the nearby visualizations is important to the user, since the user's reaction might be slower than the change of visualization. When a nice rough pattern is observed, a few successive visualizations will be similar to the observed one, allowing the user to stop ARR around the satisfactory pattern.

After a rough pattern is observed in random rendering, we switch the automated rendering from ARR to ADDR for further refinement. In ADDR, for each dimension i, α_i is continuously changed between [-1,1] by steps. Namely, α_i increases by ϵ at each step from -1 to 1, and decreases by $-\epsilon$ from 1 to -1. When a more refined cluster visualization is accepted by the user, ADDR for the dimension i is stopped and moved to the next

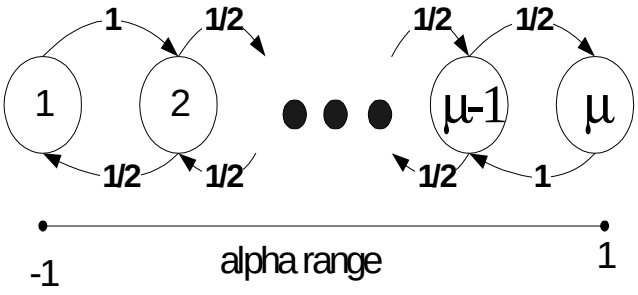

Figure 7: Markov model of random rendering of one dimension.

dimension. ARR helps to quickly find some sketch of the cluster distribution and ADDR refines the sketch to get the final visualization. Essentially, the first stage provides the main saving of time with respect to the number of interactions required to find a satisfactory sketch of the cluster distribution pattern, and is the dominating factor in determining how efficient the entire rendering will be. Therefore, we below focus on analyzing and improving the performance of the first stage.

Without loss of generality, we simplify the model as follows: each increase/decrease will move the α to certain fixed points, which is solely determined by the value ϵ. For example, if $\epsilon = 0.2$, the serial of points would be $\{$-1, -0.8, -0.6, ..., 0.8, 1$\}$. Suppose that there are μ such points, including the two endpoints -1, and 1. For simplicity, we call this set of points the μ set of points or μ points for short.

Let P_j, $1 \leq j \leq \mu$ be the probability of setting α to be one of the μ points between [-1,1]. We can model the ARR process with a Markov chain (Figure 7). It follows that $2P_1 = P_2 = \ldots = P_{\mu-1} = 2P_\mu$ [44], which implies that ARR almost uniformly sets α_i to all values in [-1, 1] (except the two endpoints, which have lower probability).

Now we define the α setting of the satisfactory sketch visualization. Suppose that a sketch of cluster distribution can be observed with the set of α_i value ranges, i.e., as long as the α_i value within the corresponding range, a satisfactory cluster visualization will be observed. We model such a subrange for α_i with λ_i = [λ_{i1}, λ_{i2}], and $|\lambda_i|$ as the number of the μ points that fall into the range λ_i. Therefore, the probability that one ARR operation finds the satisfactory sketch can be estimated by the equation 7.

$$P = \frac{|\lambda_1|}{\mu} \cdot \frac{|\lambda_2|}{\mu} \cdots \frac{|\lambda_k|}{\mu} = \frac{\Pi_{i=1}^{k} |\lambda_i|}{\mu^k} \tag{7}$$

The above equation implies two important factors in terms of the efficiency of ARR. First, the number of effective dimensions, d, is in fact less than k and may vary from dataset to dataset. As the rendering rule 1 suggests, only the "dominating dimensions" are significant to rendering. In other words, the $|\lambda_i|/\mu$ for the minor dimensions can be approximately treated as 1. Second, the individual coverage rate $|\lambda_i|/\mu$ can be increased by reducing the effective α ranges. Based on the analysis of α-adjustment (recall Section 4), smaller α_i values tend to hide the distribution detail over the dimension i, good for polishing, but the larger α values help to distinguish visual cluster overlapping. Thus, in the ARR stage, we can choose to let ARR focus on the

reduced ranges, say $[-1, -\beta]$ and $[\beta, 1]$ where $0 < \beta < 1$, for the dominating dimensions.

As observed in experiments, the rate of effective subrange λ_i to the reduced range is often quite large, and there are likely more than one $\Lambda = (\lambda_1, \ldots, \lambda_k)$ range combinations that can visualize the sketch of cluster distribution. Therefore, combined with the rendering rules, it is quite efficient to use ARR as the first step in rendering very high dimensional datasets.

However, ARR is not sufficient to find a detailed cluster visualization. A detailed cluster visualization might confine λ_is to much smaller subranges, which requires the second stage, ADDR, to refine the sketch visualization obtained by ARR. Our experiments show that by using the combination of ARR and ADDR, the cluster visualization of census dataset (68 dimensions) can be captured in around 10 minutes.

5 ClusterMap Labeling

In the labeling phase of iVIBRATE framework, we use the adaptive ClusterMap labeling algorithm to effectively extend the intermediate clustering results to the entire large dataset. The concepts of the ClusterMap and the extended ClusterMap are discussed in the paper [6]. Thus, we only provide an overview of the ClusterMap design to make this paper self-contained. We refer the readers to the paper [6] for further details.

5.1 Encoding and Labeling Clusters with ClusterMap

ClusterMap is a convenient cluster representation derived from the VISTA cluster rendering subsystem. When visual cluster rendering produces satisfactory visualization, we can set the boundaries of a cluster by drawing a visual boundary to enclose it. Each cluster is assigned with a unique cluster identifier. After the cluster regions are marked, the entire display area can be saved (represented) as a 2D byte array (Figure 8). Each cell in the 2D array is labeled by an identifier – a cluster ID (>0) if it is within cluster region, or the outlier ID (=0), otherwise. Since the size of array is restricted by the screen size, we do not need a lot of space to save it. For example, the display area is only about 688*688 pixels on 1024*768 screen, slightly larger for a higher resolution, but always bounded by a few mega pixels. As shown in Figure 8, the Cluster Map array is often a sparse matrix, which can also be stored more space-efficiently if necessary. Figure 9 is a visually defined ClusterMap of the 4D "iris" dataset. The boundaries of cluster C1, C2 and C3 were defined interactively.

In addition to the 2D array, we need also to save the following mapping parameters in Table 1 for the labeling purpose.

$Cmax_j, Cmin_j$	The max-min bounds of each column, $j = 1 \ldots k$
(x_0, y_0)	The center of the visualization
α_j	The k α parameters, $j = 1 \ldots k$
θ_j	The k θ parameters, $j = 1 \ldots k$
c	The scaling factor

Table 1: Mapping parameters for ClusterMap representation

ClusterMap representation has several advantages. First, in most situations, ClusterMap provides more details than the centroid-based or representative-point-based cluster representation. Thus, it is possible to better preserve the intermediate clustering results in the labeling phase. Second, cluster boundaries can be conveniently adjusted to adapt to any special situations or to incorporate domain knowledge as we did in the VISTA system. Finally, with ClusterMap the outliers can be better distinguished. We shall see later that ClusterMap can also be used to conveniently adapt the extension of cluster boundary.

ClusterMap representation can be applied directly in the *basic ClusterMap labeling*. It works as follows. After the ClusterMap representation is loaded into the memory, each item in the entire large dataset is scanned and mapped onto one ClusterMap cell. The mapping follows the same mapping model used in the visual rendering system. Suppose that the raw large dataset is stored on disk in form of N-row by k-column table. We rewrite the mapping formulas as follows:

$$\text{Normalization: } x'_{ij} = \omega_j * (x_{ij} - Cmin_j) - 1 \tag{8}$$

$$\omega_j = 2/(Cmaxj - Cminj)$$

$$\alpha\text{-mapping: } x_i = \sum_{j=1}^{k} \psi_x(j) x'_{ij} - x_0, y_i = \sum_{j=1}^{k} \psi_x(j) x'_{ij} - y_0 \tag{9}$$

where $\psi_x(j) = c\alpha_j \cos(\theta_j)/k$, $\psi_x(j) = c\alpha_j \sin(\theta_j)/k$ and ω_j can be pre-computed, and other parameters, such as c and θ_j are the same as defined in VISTA visualization model.

Sequentially, the algorithm reads the i-th item $(x_{i1} \ldots x_{ik})$ from the k-D raw dataset, normalizes it and maps it to a 2D cell coordinate (x_i, y_i). From the cell (x_i, y_i), we can find a cluster ID label, or a outlier label, depending on the ClusterMap definition.

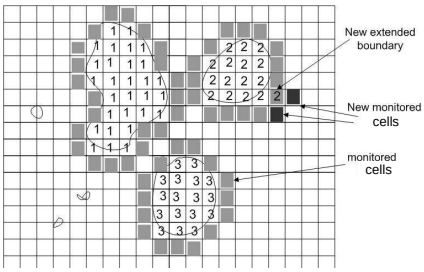

Figure 8: ClusterMap with the monitored area where $\epsilon = 1$

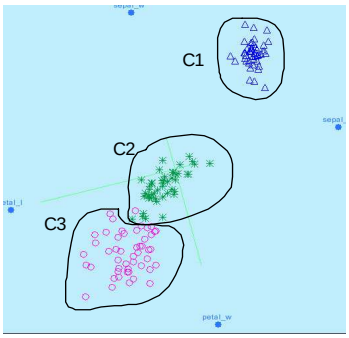

Figure 9: ClusterMap of the 4D "iris" data

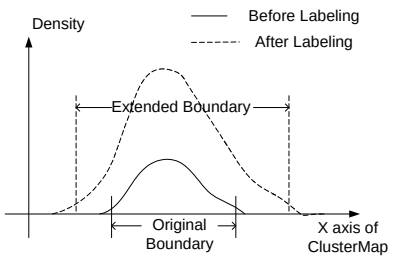

Figure 10: Boundary extension – the cross section of a typical evolving cluster in ClusterMap.

5.2 Adaptive ClusterMap for Boundary Extension

In the basic ClusterMap labeling, we assume that the cluster boundary defined on the sample set will not change significantly during the labeling process. However, with the increase of labeled items, the density in the original boundary areas will increase as well. Thus, the original boundary defined on sample set shall be extended to

some extent. An example has been shown in Section 1.2 (Figure 2). Boundary extension encloses the nearby outliers into the clusters and may require the mergence of the two nearby clusters if they become overlapped. Figure 10 sketches the possible extension in ClusterMap with the attending of labeled items, in terms of the point density.

Boundary extension is maintained by monitoring the point density around the boundary area. We have the initial boundary defined in ClusterMap representation. We name the cells within the cluster boundary as the "cluster cells" and the cells around the boundary area as the "boundary cells". The initial boundary cells are precisely defined as within a short distance ε away from the initial boundary. All non-cluster cells are "outlier cells" including the boundary cells. We define the density of a cell on the map as the number of items being mapped to this cell. Apparently, the density of boundary cells should be monitored in order to make decision on boundary extension. A threshold density, δ, is defined as two times of the average density of outlier cells. If the density of a boundary cell grows to δ with the attending of labeled items, the boundary cell is turned into a cluster cell and results in the extension of boundary $-$ The non-cluster cells within the ε-distance from the old boundary cell become the new boundary cells. Since the boundary is on the 2D cells, we can use cell as the basic distance unit and the "city block" distance [52] as the distance function to define the ε-distance. ε is often a small number, for example, 1 or 2 blocks from the current boundary.

To support the above adaptive algorithm, we need to extend the basic structure of ClusterMap. First, for each cell, we need one more field to indicate whether it is a monitored non-cluster cell or not. We also need to keep track of the number of points falling onto each cell. This information is saved at a "Density Map". In addition, δ should be periodically updated according to the average noise level, since the average noise level will also rise with the increase of labeled items. For detailed discussion of the ClusterMap algorithm, please refer to the paper [6]

The Adaptive ClusterMap labeling algorithm can be performed in two scans: The first scan generates an extended ClusterMap and the second scan can be performed to build up a R-tree index on the map for efficient access to the items on disk. After the first scan, the adjusted ClusterMap can also be checked with the ClusterMap Observer to identify the anomalies (section 6.2). The second scan is very helpful for many clustering applications that involve similarity search [37] and cluster-based indexing, which require efficient access of the cluster members.

5.3 Complexity of Labeling Algorithms

The two key factors that measure the effectiveness of the labeling algorithms are *accuracy* and *computational cost*. Although ClusterMap brings apparent advantages in describing precise cluster boundaries, accuracy will be further evaluated in experiments. We analyze the other important factor: the computational cost in this section for the four labeling algorithms: CBL, RPBL, Basic ClusterMap, and Adaptive ClusterMap.

One way to estimate the cost of a labeling algorithm is to count the number of necessary multiplications. For example, one k-D Euclidean distance calculation costs $O(k)$. Based on the formulas 8 and 9 given in section

Algorithm	Complexity	LDS data (N=1M, k=5, r =5)	Census data (N=1M, k=68, r=3)
CBL	[$kN \log_2(r)$, rkN]	4.67	56
RPBL	[$kN \log_2(rm)$, rmkN]	6.5	65
Basic ClusterMap	3kN	4.42	54
Adaptive ClusterMap	$\sim$ 6kN	8.69	108

Table 2: Cost estimation of the four algorithms

5.1, we can roughly estimate the cost of the basic ClusterMap labeling. Map reading and parameter reading cost little constant time due to the limited small map size. For each item in the dataset, max-min normalization costs $O(k)$ as shown by Formula 8. α-mapping function costs $O(k)$ to calculate the x and y coordinates with Formula 9, respectively. Locating the cell in ClusterMap to get the corresponding cluster ID costs constant time. Hence, the total cost for the entire dataset is $O(3kN)$, where N is the number of rows in the dataset. While the Adaptive ClusterMap runs with the two scans, the cost is roughly two times of the Basic ClusterMap labeling, i.e., $O(6kN)$.

When kd-tree [18], or other multi-dimensional trees, is used to organize the representative points or centroids, we get the near-optimal complexity for the distance-comparison based labeling algorithms. Let r be the number of clusters and m be the number of representation points per cluster. The cost to find the nearest neighbor point in kd-tree is at least $\log_2(rm)$ distance calculation for RPBL or $\log_2(r)$ for CBL. For a typical RPBL as reported in the CURE paper, the number of representative points has to be greater than 10 ($m \geqslant 10$) in order to roughly describe the regular non-spherical cluster shapes (mainly, the elongated shapes). The number should increase substantially if the irregular cluster shapes are detected. Conservatively, the cost of RPBL will be at least $4kN$, normally a little higher than that of the basic ClusterMap. The cost of CBL should be around $\log_2(r)kN$ or rkN if r is small and a tree structure will increase the cost.

Both CBL and RPBL need a small amount of memory, $O(rk)$ and $O(rmk)$, respectively. Let w be the width and h be the height of the 2D ClusterMap, the basic ClusterMap will need $O(wh)$ memory, which counts for several megabytes in practice. Correspondingly, the adaptive ClusterMap needs about $O(2wh)$ memory.

Table 2 summarizes the formal analysis. The cost on two large datasets, LDS and Census data, which will appear in the later sections, are also listed in Table 2 to give a feeling of the real cost. For both datasets, m is set to 20 and the time unit is second.

In summary, the Adaptive ClusterMap labeling algorithm uses a little more time and space to label the datasets but this small extra cost can bring huge benefits as we will show in the following sections.

6 Integrating the Three Phases

Integrating the three phases (Sampling, Visual Cluster Analysis, and Adaptive ClusterMap Labeling) under the iVIBRATE framework presents some interesting and unique challenges. Since the phases are interconnected in sequence, without proper operations in the earlier phases, errors could be propagated and aggravated in the later phases. In this section, we investigate two important issues in integrating the three phases. First, we study the

effect of the sampling phase on the later two phases, primarily the impact on determining the max-min bounds from samples (section 6.1) and exploring the small clusters hiding in outliers (section 6.3). Second, we analyze the possible influence of Visual Cluster Analysis on the labeling phase, and develop some anomaly monitoring methods to control and reduce the errors (section 6.2).

6.1 Determine the Max-min Bounds from Samples

The VISTA subsystem requires to determine the max-min bounds for normalization, denoted by "$Cmax_j$" and "$Cmin_j$". These bounds are used not only in the rendering phase by the α function but also in the labeling phase by the ClusterMap algorithms. Thus, these bounds should be kept unchanged throughout the three-phase clustering process. Max-min normalization is the first step in the VISTA visualization model (Section 4.1), which prepares the data for α-mapping without loss of any information for visual cluster rendering. However, since the max-min bounds are obtained from the sample set, they may differ from the actual bounds for the entire dataset. Inappropriate setting of bounds may cause additional errors in both the clustering phase and the labeling phase.

Concretely, the effect of inappropriate bounds is twofold. First, if the max and the min bounds are too tight (i.e., the two bounds are too close to one another), even though they enclose all samples, there might be high out-of-bounds rates for the entire large dataset, which increases the amount of errors generated at the labeling phase. On the other hand, if the max-min bounds are too loose, most values are scaled down to a narrow range and the difference between the values cannot be observed efficiently in cluster rendering. Recall that the α-mapping in equation 2 of Section 4.1 shows two possible ways that we can adjust the visual parameters in order to observe the visual difference between different values: one is to adjust the α_i values ($1 \leq i \leq k$) and the other is to alter the scaling factor c. Since the α values ($\alpha_1, \ldots, \alpha_k$) are restricted in the range of [-1, 1] for the purpose of efficient interactive rendering, we might have to adjust the scaling factor c to a large value, which, however, could improperly enlarge the entire visualization and leave some part of visualization out of the display area. Therefore, the ideal bound setting will be located at a narrow range.

The first problem is how large the sample bounds can work approximately as the overall bounds. We address the problem by studying the relationship between the sample value bounds and the sample size – if we just use the sample value bounds as the overall max-min bounds, how many sample points do we need in order to find the bounds that are also appropriate for the entire dataset, i.e., enclosing almost all points? The problem of bounds estimation based on the sample data can be formalized as follows.

Let n denote the size of the sample dataset and p denote the probability of points in the entire dataset enclosed by the sample bounds. Since bounds estimation for each column is independent, without loss of generality, we can treat the values from one column as samples of a random variable X. We now estimate the bounds for the random variable X with the sample set, so that the bounds cover $100p$ percent of the distribution of X with certain confidence level. This problem can be exactly modeled as *Tolerance Interval* [10].

Definition 1. *A tolerance interval* $(r \leqslant X \leqslant s)$ *with tolerance coefficient* γ *is a random interval. Its range* $[r$*,*

s] includes at lease $100p$ *percent of distribution with the probability* γ.

In our case, we fix the two end points as the two order statistics, $X_{(1)}$ and $X_{(n)}$, i.e. the max and min values of the sample set, for easy processing. The above definition then can be rephrased as:

$$P[P(X_{(1)} < X < X_{(n)}) \geqslant p] = \gamma \tag{10}$$

Let $F_X(x)$ be the distribution function of X and $U_{(n)}$ and $U_{(1)}$ be the max and min values of n uniform samples in [0, 1]. $P(X_{(1)} < X < X_{(n)})$ is equal to $F_X(X_{(n)}) - F_X(X_{(n)}) = U_{(n)} - U_{(1)}$. Therefore, without knowing the distribution of X, we can find the distribution of $U_{(n)} - U_{(1)}$ instead, which is solely related to the order statistics of uniform distribution. Let $U = U_{(n)} - U_{(1)}$, it is easy to find the joint distribution with order statistics. We can get the density distribution of $U = U_{(n)} - U_{(1)}$, $f_U(u, v)$ as follows.

$$f_U(u, v) = n(n-1)u^{n-2}(1-u) \tag{11}$$

Now, since $\gamma = P(U \geqslant p)$, we can get the following relation between γ, p, and n.

$$\gamma = \int_p^1 n(n-1)u^{n-2}(1-u)du \tag{12}$$

The right side of the equation is the *incomplete Beta function* (represented as $betainc(1-p, 2, n-1)$ in Matlab). Fixing one of the three parameters γ, p, and n, we can infer the relation between the other two parameters. We are more interested in the range of the sample size for a large p so that the sample bounds can cover almost all points in the entire dataset. By setting p to a very high probability, 0.999, we find the relationship between γ and sample size n as Figure 11 shows.

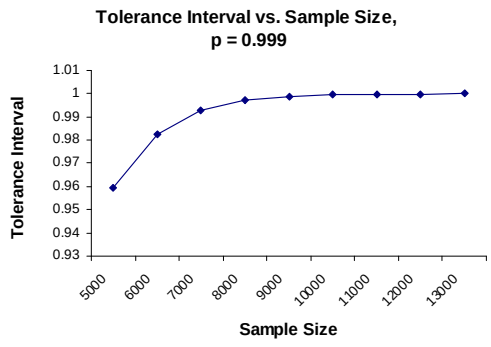

Figure 11: The relation between γ and n, $p = 0.999$

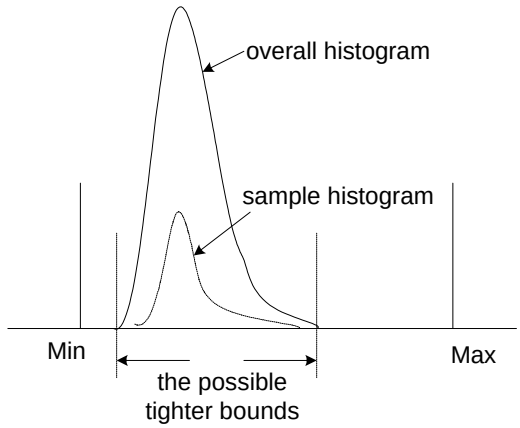

Figure 12: The possible tighter bounds for skewed distribution

At sample size $n \approx 13,000$, the tolerance level almost reaches 100%, which means that we can confidently say that the max-min value bounds of a sample set in size of $n = 13,000$ or larger are the bounds which cover

99.9% records in the entire dataset. Note that the number 13,000 is induced without any assumption about the data distribution and the sample size for real datasets. For real datasets that have some special distributions, the sample size should be smaller as shown in section 8. Therefore, we consider this as the upper bound of sampling size.

Our experiments with the first prototype of iVIBRATE shows that its VISTA cluster rendering subsystem can comfortably handle up to 50,000 items with 30-50 dimensions in near real time, in a common computer system environment (for example, CPU 1.5Ghz, memory 256M) [7]. Thus, the VISTA subsystem can also comfortably render a large sample set ($n \approx 13,000$), which definitely contains the max-min bounds for the entire dataset.

The second problem is that the initial max-min bounds based on the sample value bounds might also be too wide, when the distribution is skewed as shown in Figure 12. In a skewed distribution, almost all points are located within a narrow range, with small amount of points far away from the center. This can frequently happen in most real datasets. In this case, if we simply use the sample bounds for normalization, the cluster rendering subsystem may not work efficiently as we have discussed. This can be checked by the histogram of the sample data column. If the skews are found in the sample dataset, we may need to check the histogram for the entire dataset to carefully narrow down the bounds. Based on the above analysis, we suggest the following steps to choose the normalization bounds for each column.

1. Sample the dataset to get a sample set in size around 13,000;
2. Find the max-min bounds of the sample set as the initial bounds for each column and build the histograms for the columns with the sample set;
3. If some columns have very skewed distribution with a few outliers, we build the histograms for these columns from the entire dataset. The loose bounds can be narrowed down according to the histograms for entire dataset, while maintaining the out-of-bounds rate as $1 - p$, e.g., p=0.999.

By doing this, we can avoid the relatively expensive third step for some datasets. However, in the worst case, the overall cost of finding the proper bounds is still quite acceptable. If no skew is found in the second step, the total cost is $O(n)$, where n is the sample size. Otherwise, it is $O(N)$, where N is the size of the large dataset. Since this is a one-time process, $O(N)$ is still not bad.

6.2 Monitoring the Anomalies in ClusterMap Labeling

Boundary extension can behave abnormally due to low sample rate, imprecise rendering result, or inappropriate setting of the initial cluster boundary. We first discuss two possible anomalies in the ClusterMap labeling process, and then introduce the methods to monitoring and handling these anomalies.

- The first anomaly is the vague cluster boundary. The cluster boundary becomes vague soon after labeling certain amount points, while the normal boundary extension should be slow and happen uniformly around the boundary throughout the entire labeling process. There are two situations that can cause such a

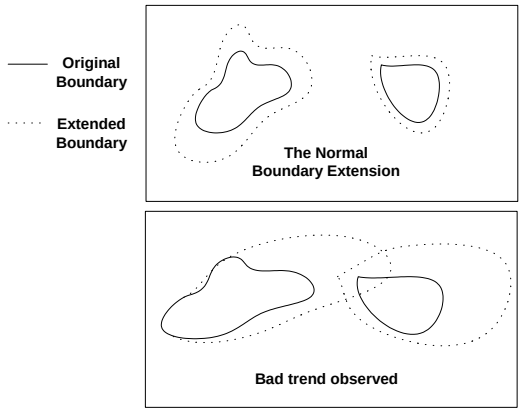

Figure 13: Anomalies that require fine adjustment of α values

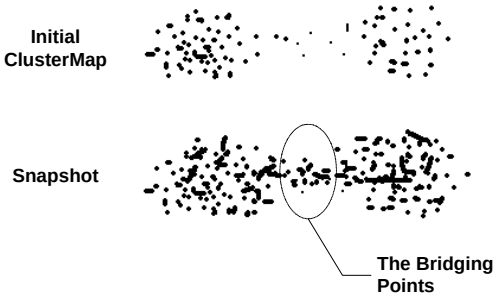

Figure 14: The bridging points

anomaly, illustrated by Figures 13 and 14. First, the initial distances between the clusters are not defined appropriately due to the sample size or lack of visual refinement, which requires the user to tune the initial ClusterMap, e.g., adjusting the α parameters slightly in VISTA. This is illustrated by Figure 13. Second, the other situation is the "bridging points" between the clusters, which are not dense enough in the sample set but they may form the "bridge" that connects the clusters later in the labeling process, as shown in Figure 14. The user has to make decision based on the domain knowledge to either split or merge them.

- The second anomaly is that the ordering of data records on disk may affect the boundary extension. For example, a sequence of data is mapped to a focused boundary area at early stage of labeling thus the a false boundary extension occurs. However, later on no more points are mapped to that area. As a result, this area is falsely extended as a part of cluster. We observe that this error only happens in the situation where such particular data records are stored together and the labeling is done according to the original ordering of data records on disk. This anomaly can be avoided by accessing the data records in a perturbed sequence. We use a method named "sequence perturbation" (Figure 15). To put it simply, if the large data file is regarded as a block file, we equally divide the dataset into s sequences of blocks. In each processing window, we read some data blocks at the head of each sequence and perturb the ordering of the records in these blocks. This can almost eliminate the risk of non-uniformity in data ordering.

In general, these anomalies can be monitored with the "snapshots" of labeling, which are visualized by the tool "ClusterMap Observer". Snapshots are a series of evolving ClusterMaps and density maps, which incorporate the boundary extension and are saved at some time interval during the first scan of Adaptive ClusterMap labeling. The user can observe the snapshots with ClusterMap Observer. If the anomalies are observed, the user can terminate the labeling process early and returns to VISTA subsystem to adjust the original ClusterMap. Figure 16 shows a snapshot with 10 million records labeled. The noisy areas around cluster A and B are not labeled as cluster cells since the density of these cells does not reach the threshold. Whether these cells should be included into clusters or not, may depend on the user's requirement. However, the extended boundary can always be edited with ClusterMap Observer, which makes the entire labeling process very flexible and manageable.

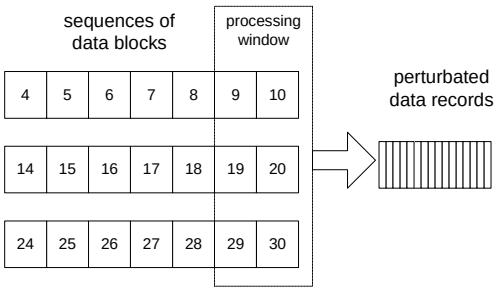

Figure 15: Record perturbation

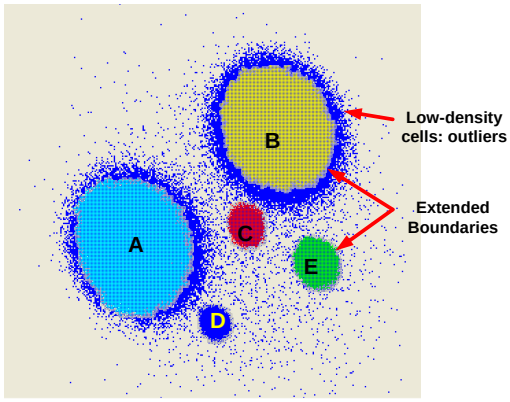

Figure 16: A snapshot of labeling a large dataset, visualized with ClusterMap Observer

6.3 Detect and Explore the Small Clusters

Missing small clusters is most likely caused by low sample rate, e.g. less than 1% of the entire dataset. The small clusters may start to emerge as the labeling proceeds, which could be detected in the snapshots of labeling.

If there are small clusters emerging, we can use the following filtering method to confirm and explore the small clusters in detail. In the labeling phase, we run the Adaptive ClusterMap labeling to label all records, and then extract the outliers *only* from the large dataset for visual rendering. If the outlier dataset is still large, it is sampled and rendered in VISTA cluster rendering system again. Since the size of the outlier dataset is usually much smaller than that of the original dataset, one additional sampling for the outlier dataset is often sufficient to discover the small clusters in it. Similarly, the observed small clusters are marked in an *additional* ClusterMap. We can repeat this process until the size of the outliers becomes negligibly small. This process might result in a couple of additional ClusterMaps representing the small clusters at different detail levels. These ClusterMaps are used together to effectively label the interested small clusters.

In this process, the user can always control the “drill-down” level and the size of interested small clusters. More flexibly, the user can select any interested area of ClusterMap and zoom in to observe the possible small clusters in the corresponding portion of data only. This can be done iteratively, which results in a general extended iVIBRATE framework for hierarchically exploring the clusters in very large datasets (Figure 17). In short, under the iVIBRATE framework, users have more flexibility in monitoring and exploring the details in clustering structure. To our knowledge, no one of the existing approaches has provided such flexibility.

7 EXPERIMENTS

This section presents three sets of experiments. The first set of experiments shows the effectiveness of VISTA visual clustering rendering in finding irregularly shaped clusters. The second set of experiments shows that ClusterMap labeling can handle outliers and irregularly shaped clusters with low computational cost. The third

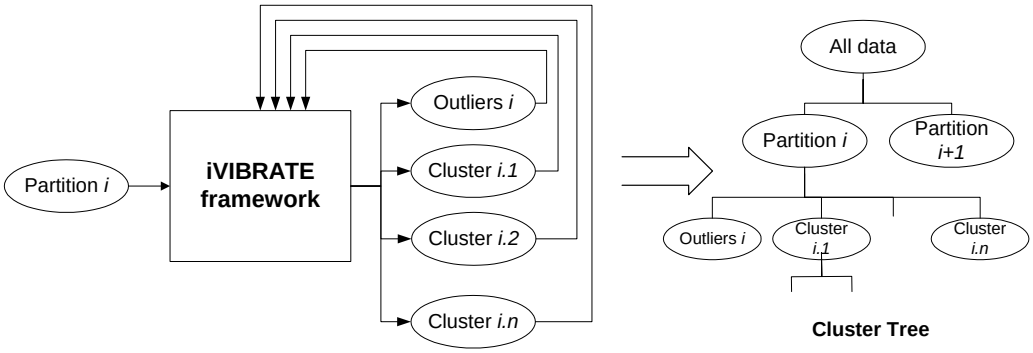

Figure 17: Iterative exploration in the extended iVIBRATE framework

set of experiments demonstrates the advantage of Adaptive ClusterMap labeling. The results show that this visualization powered framework iVIBRATE is more reliable and flexible than any existing approaches.

7.1 Datasets and Experiment Setup

The first set of experiments are conducted on a number of well-known datasets that can be found in UCI machine learning database [1]. These datasets, although small or median in size, have irregular cluster distribution, which is an important factor for testing the effectiveness of the VISTA system. We carefully choose these datasets with the following three factors in mind: 1) the current version of VISTA system only concerns the datasets having a manageable number of numerical attributes; 2) clusters in most of the datasets are not in regular spherical shape, the size of cluster may vary greatly, and the distance between clusters can be so close that the algorithmic approaches can easily fail to distinguish; 3) the existing class labels can effectively indicate the irregular clusters. For easy comparison, we also ignore the tiny clusters in some datasets, for example, in "ecoli[2]" data and "shuttle" data.

Dataset	N	# of dim.	# of clusters
Breast-w	699	10	2
Crx	690	15	2
Ecoli	336	7	8
Hepatitis	155	19	2
Ionosphere	351	34	2
Iris	151	4	3
Wine	178	12	3
Shuttle.test	14500	9	7

Table 3: The datasets used in visual rendering.

Two datasets are used for the second set of the experiments. One is the simulated dataset DS1 used in CURE [21]. DS1 is a 2D dataset having five regular clusters, including three spherical clusters, two connected elliptic clusters, and many outliers. In our experiments, DS1 is used to evaluate the effect of outliers on the labeling algorithms. The second dataset is the "shuttle" dataset (STATLOG version, test dataset) introduced in the first

[1] http://www.ics.uci.edu/∼mlearn/Machine-Learning.html

[2] There are totally 8 attributes in Ecoli data, but one is the name of E.Coli, which is discarded in clustering.

set of experiments. It is a 9-dimensional dataset with very irregular cluster distribution. There are seven clusters in this dataset, among which one is very large with approximately 80% of data items, and two are moderately large with approximately 15% and 5% of data items, respectively. Others are tiny clusters and thus ignored in comparison. "Shuttle" dataset is used to evaluate the effect of irregular clusters on the labeling process. These two datasets should show how ClusterMap avoids the common problems of the traditional labeling algorithms.

In the third set of experiments, a simulated 5-dimension large dataset LDS with one million records is designed to test the performance of Adaptive ClusterMap on very large datasets. Figure 26 shows a 10K sample set visualized with VISTA system. LDS simulates 5 clusters – three are approximately spherical, and the other two are in irregular shape. There are also about 1% outliers. LDS is well-designed so that we can approximately predefine the control labels for the entire dataset with small errors. This dataset is used to evaluate the effect of all three factors: outliers, irregular clusters, and boundary extension.

The three labeling algorithms, CBL, RPBL, and ClusterMap are implemented in C++. RPBL is based on the boundary points generated by the CURE clustering algorithm, which was known as a fine RPBL adapted for non-spherical cluster. We run CURE clustering to get the boundary points with the following parameters: the number of representative points is 20 and α (the shrink factor) is set to 0.5 as suggested. We also use ANN (Approximate Nearest Neighbor) C++ library from University of Maryland at College Park to construct kd-trees for RPBL and CBL in order to improve the performance of nearest neighbor search.

7.2 Visual Cluster Rendering

In this section we will introduce the experimental result concerning the power of visual cluster rendering system in finding clusters. The VISTA visual clustering system was implemented in Java [3].

When we finish interactive cluster rendering, we mark the cluster areas, in which the points are respectively labeled with the cluster ID. With the original labels in the datasets, we can define the items that are wrongly clustered as the errors, the number of which divided by the size of the dataset is the error rate of visual cluster rendering result.

First, we use unguided visual rendering (UGV), where no external information is incorporated, to find the visual partition. Unguided visual rendering only depends on the visually observed dense-point areas and the gaps between the areas to discern the clusters and the cluster boundaries. Since there is visual bias on the visualization, the visual rendering sometimes may trap in local minima, where some visual cluster overlaps are not distinguished. It has also been shown that solely depending on visual information to define the clusters is error-prone [11, 25].

We could possibly avoid the local minima by incorporating some external information, either from the result of clustering algorithms or from the domain knowledge. This results in the second rendering method "guided visual rendering (GVR)". In our experiment, ten of labeled items from each cluster are randomly selected serving as the "landmarks" in GVR.

[3] http://disl.cc.gatech.edu/VISTA

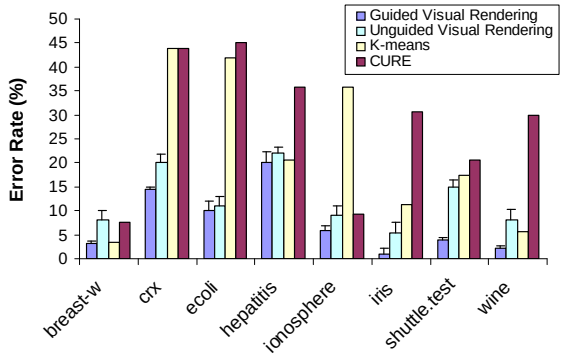

Figure 18: Comparison of error rates on the experimental data

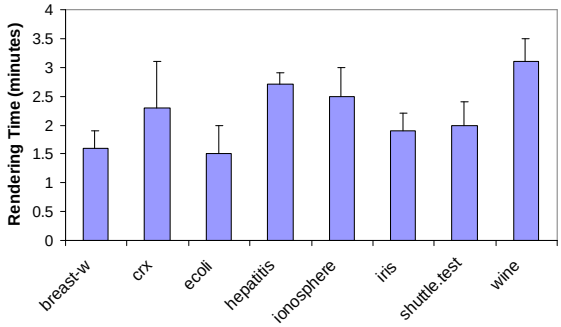

Figure 19: Visual rendering cost (GVR) on the experimental data

We also compare the visual rendering with two algorithms, K-means and CURE algorithms to see the possible improvement. CURE clustering [21] is recognized as one that can deal with irregular cluster shapes in some degree, and K-means is the most popular algorithm identifying only spherical clusters. Both algorithms use the normalized data as input. The K-means results shown in Figure 18 are the best result in ten runs.

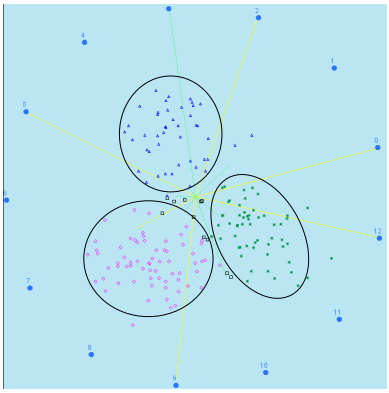

Figure 20: Visualization of "wine" data

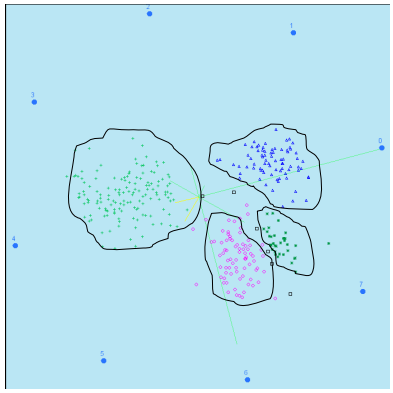

Figure 21: Visualization of "Ecoli" data

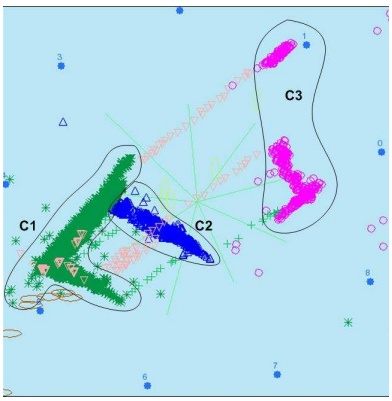

Figure 22: Visualization of "shuttle" data

The experimental result shows that neither CURE or K-means can deal irregularly shaped clusters very well, and UGV may also trap in some local minima for some datasets, for example, "breast-w (breast-cancer-wisconsin)" and "wine" (Figure 20). However, by combining the limited external information we can improve the UGV result to some extent. For example, "Iris" and "Ecoli" (Figure 21) datasets have very clear cluster structure thus the UGV and GVR yield almost the similar results. But in rendering "shuttle" (Figure 22), we need the help of the landmark points to distinguish the clustering structure, as the domain-specific clusters are formed by merging or splitting some irregular clusters.

In addition, the average interaction time (GVR) of five trained users (Figure 19) indicates that it is not difficult for a trained user to find a satisfactory visualization with the help of the visual rendering rules in section 4.2. More detail user studies will be performed to evaluate the visual design.

7.3 Outliers and Irregular Cluster Shapes

We have discussed three different labeling algorithms: Representative-Point Based Labeling (RPBL), Centroid-Based Labeling (CBL), and ClusterMap (basic ClusterMap in this set of experiments). In this section we study the performance and error rates in labeling the outliers and irregularly shaped clusters.

We run VISTA to get the ClusterMaps in the map resolution of 688×688. The cost to rebuild a ClusterMap structure in memory is about 340$\sim$360ms, which can be ignored for processing very large datasets. Both DS1 and shuttle datasets show that the estimation of cost is appropriate – all three algorithms are linear complexity and the basic ClusterMap is almost the fastest one (Figure 23).

The DS1 dataset is used to show the effect of outliers on the algorithms. The result shows that the error rate of ClusterMap is much lower than the other two algorithms. By visualizing the labeling result, we observed that CBL can suffer from the variant cluster sizes, the distance between clusters, and the outliers. Particularly, we take a look at the visualization of RPBL result (Figure 24), which clearly shows that the outliers are labeled as the members of the nearby cluster and some points on the cluster boundary are incorrectly labeled too.

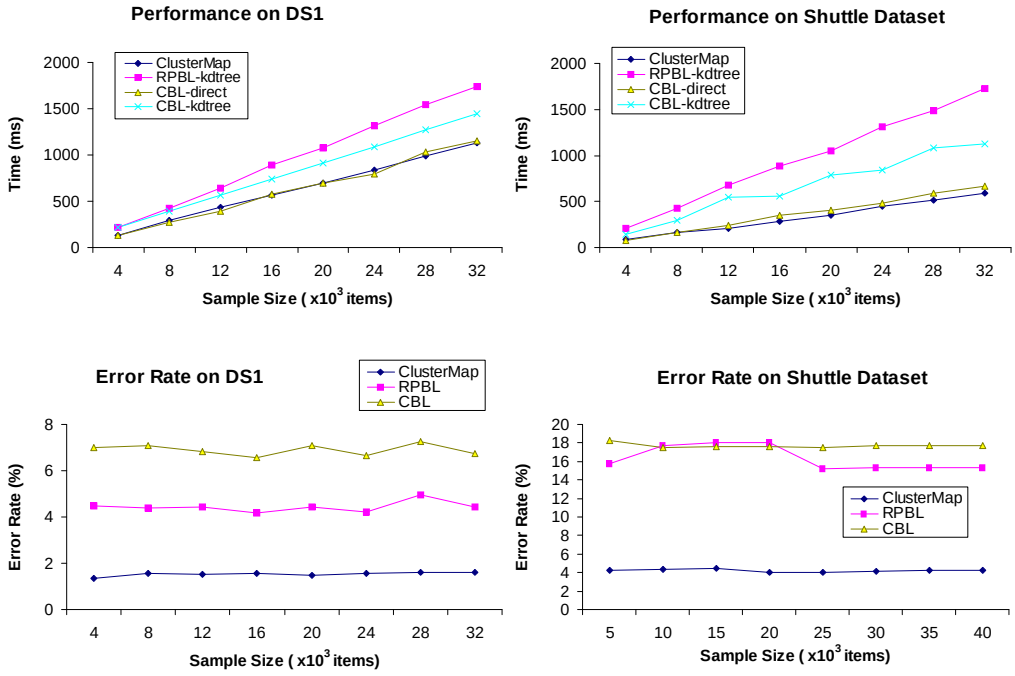

Figure 23: Labeling outliers and irregularly shaped clusters

"Shuttle" dataset has very irregular clusters. Without the incorporation of domain knowledge, the intermediate clustering will not be satisfactory. If not impossible, it is very difficult for the automated clustering algorithms to incorporate such important external information. We use the shuttle dataset to show that the error in intermediate clustering result might be amplified in labeling phase with the existing labeling algorithms. With the increasing number of labeled records, the error rate of RPBL increases to the level similar to that of CBL, due to its lack of ability dealing with the very irregular cluster shapes. The basic ClusterMap labeling keeps consistent

with the VISTA cluster rendering result and thus has much lower error.

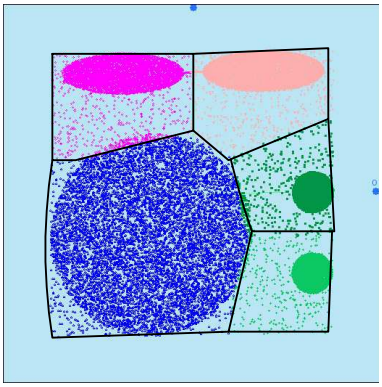

Figure 24: Outliers are labeled as the members of the nearby clusters

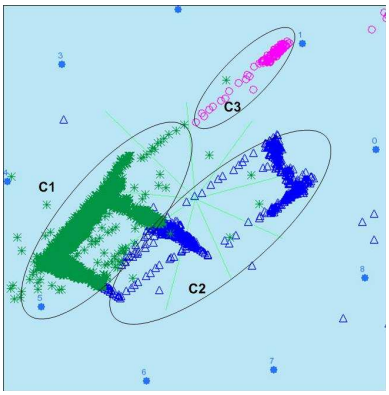

Figure 25: Visualization of the inaccurate RBPL result on Shuttle data

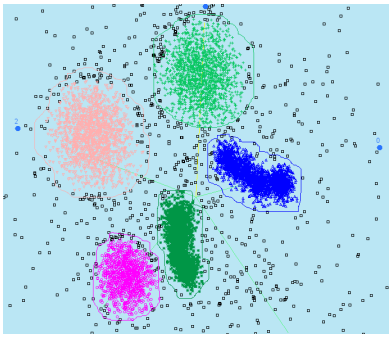

Figure 26: Visualization of 10K samples of LDS

7.4 Adaptive ClusterMap on Large Dataset

This experiment on the large dataset LDS mainly shows the scalability and effectiveness of Adaptive ClusterMap algorithm, especially in dealing with outliers and boundary extension.

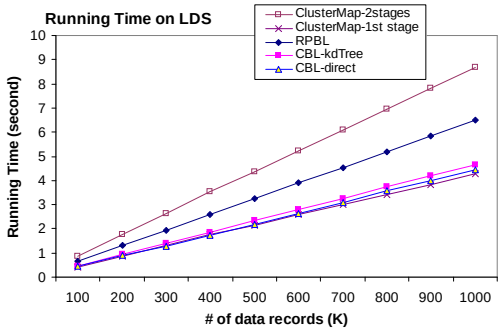

Figure 27: Performance on LDS

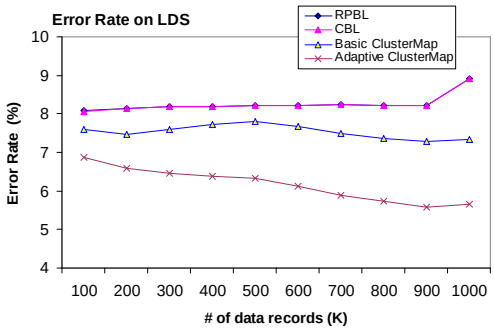

Figure 28: Error rate on LDS

The progressive plots of time cost and error rate are shown in Figure 27 and 28. The performance curve shows that the two-stage cost of Adaptive ClusterMap labeling will be a little greater than the other algorithms, however, it still keeps linear in terms of the number of labeled records. If the second stage is not necessary for some applications as we discussed, the first stage will only cost as little as CBL. Therefore, the Adaptive ClusterMap labeling is still scalable to the number of records.

From the error rate curves (Figure 28), we can observe the difference and the trend between the four algorithms. Basically, RPBL and CBL have higher error than the ClusterMap labeling algorithms due to the lack of ability dealing with outliers and imprecise boundary definition. Since the clusters are well-separated, as the number of labeled records increases, RPBL and CBL basically have the similar labeling results and thus the error rates are very close (they are overlapped in the figure). The basic ClusterMap does not consider the boundary extension,

therefore, has higher error than Adaptive ClusterMap. The error of Adaptive ClusterMap tends to decrease with the increasing number of labeled records, because the more the labeled records are incorporated,the better the Adaptive ClusterMap labeling can approximate the real cluster boundary.

8 Exploring Clusters in Very Large Census Dataset: a Comprehensive Example

In this section, we analyze the clusters in a real large dataset – the 1990 Census dataset using the iVIBRATE framework. The procedure revisits most of the concepts, methods, and models we have presented in the previous sections. Concretely, it includes max-min bounds checking and refining in the sampling phase, rendering the sample dataset with the VISTA subsystem, using adaptive ClusterMap labeling to label the entire dataset, and analyzing the small clusters missed by sampling. Let's start with the description of the dataset.

8.1 Dataset

The very large "US Census 1990 Data[4] " is used in this empirical study to show the effectiveness and flexibility of the iVIBRATE framework. This dataset is a discretized version of the raw census data, originally used by the paper [40] in studying the relationship between the sampling approach and the effectiveness of Expectation-Maximization (EM) based clustering algorithms for very large datasets. This dataset is very large in terms of both the number of records and the number of attributes. Although many of the less useful attributes in the raw dataset are dropped, the total number of preserved attributes still reaches 68. It also contains more than 2 million (2,458,284) records, about 352 megabytes in total. Since the dataset is a discretized version, we also run an entropy-based categorical clustering/validation algorithm (ACE algorithm for finding a sub-optimal partition and BkPlot for determining the best Ks) [8] to cross-validate the result in the phase of iterative cluster analysis.

8.2 Sampling and Bounds

In section 6.1, we have formally analyzed the out-of-bounds rate in terms of the sample size. Note that the analysis is good for any data distributions, which actually results in a quite conservative estimate to the appropriate sample size (around 10,000 sample records) so that the sample bounds are approximately the global bounds. In order to verify this analysis, we try two sets of sample datasets with 5K records and 10K records, respectively. Each set consists of 10 sample datasets, generated by uniformly sampling the entire census dataset. By checking the bounds, we find only two 5K-record sample sets having 1 and 2 attributes, respectively, missing the maximum discretized values, which cause less than 0.1% out-of-bounds rate for the entire dataset. All of the 10K-record datasets include the global bounds. This demonstrates that the estimated threshold in section 6.1 is indeed a conservative number. By checking the histograms, we confirm that it is unnecessary to adjust

[4]In UCI KDD Archive http://kdd.ics.uci.edu/databases/census1990/USCensus1990.html

the initial max-min bounds for effective rendering. Therefore, both 5K-record and 10K-record sample datasets should be fine for effective visual rendering.

8.3 Visual Cluster Rendering

Due to the high dimensionality, manually rendering the data with the dimension-by-dimension method is not recommended for the census dataset. The semi-automated rendering method: automatic random rendering (ARR) followed by automatic dimension-by-dimension rendering (ADDR) (section 4.3) is used in rendering a set of 10K-record sample datasets.

Recall that ARR is a randomized process, visual rendering with different rounds of ARR can result in different visualizations. We conduct visual rendering 5 times to see the difference between the rendering results. By comparing the ARR results, the two visually well-separated clusters are confirmed as the same clusters in all of the five renderings. We list some numbers in Table 4 reflecting the consistency between the visualizations. The rendering time is the wall-clock time with the unit of half minute. Due to the high dimensionality and dense point clouds, the average rendering time is about 10 minutes, much higher than those shown in Figure 19. Automatic random rendering (ARR) can quickly identify the sketch of cluster distribution (2 clusters) in about 2 minutes, while automatic dimension-by-dimension rendering (ADDR) should take longer time to refine the visualization. In Table 4, only the number of records in C1 and C2, are shown, the rest of the 10K records are regarded as outliers. ”Shared” represents the shared percentage of the points between the current round of rendering result and the other rounds.

No.	Random Rendering (minutes)	Automatic DDR (minutes)	C1 (pts)	C1 Shared (%)	C2 (pts)	C2 Shared (%)
1	2.0	8.0	2001	93.0%	7546	87.4%
2	1.5	9.0	2198	84.7%	7610	86.7%
3	2.0	7.0	2070	89.9%	7252	90.9%
4	2.5	8.5	2040	91.2%	7403	89.1%
5	1.5	7.0	2189	85.0%	7508	87.8%

Table 4: Rendering the 10K-record sample census dataset.

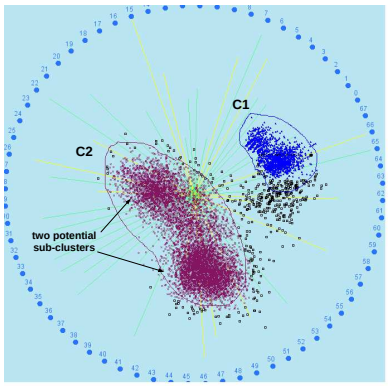

Figure 29: Result of visual rendering 1

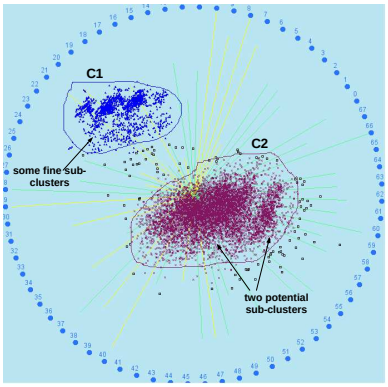

Figure 30: Result of visual rendering 2

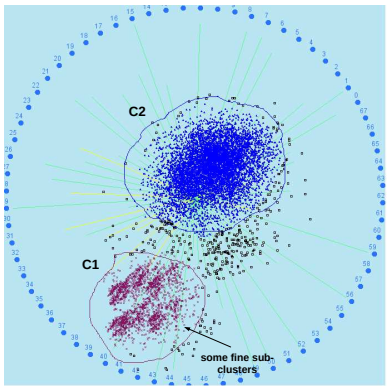

Figure 31: Result of visual rendering 3

We show three of the five visualizations in Figure 29, 30, and 31. All three results clearly show the two major

clusters. Figure 29 and 30 also show that C2 could potentially have two subclusters. Cluster validation with the entropy criterion [8] shows that the "best K" for clustering census dataset should be 3 or 2 (Figure 34), which implies that the initial visual clue about the possible existence of three clusters could be true. Similarly, Figure 30 and 31 show that C1 may have some fine structure, which we can also further refine if necessary.

By exploring the clusters C1 and C2 separately, we identify that C2 indeed contains two sub-clusters. We do not show the separated rendering result but the refined visualization of the three clusters only (Figure 32). The result is confirmed by the entropy-based cluster validation method. Figure 33 shows that the visually separated clusters match well with the clusters labeled by the entropy-based ACE categorical clustering algorithm [8]. Note that from the algorithmic result, we are not able to identify the outliers and the initial cluster boundary, but with the VISTA system we are able to interactively define the initial cluster boundary. Figure 32 also shows the initial cluster boundary which is used to generate the ClusterMap for the labeling phase.

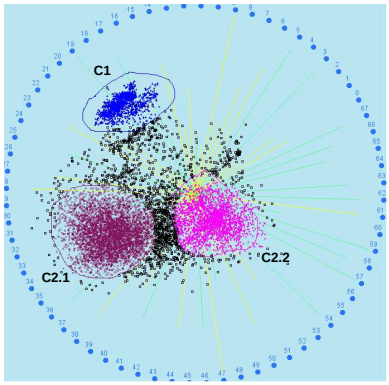

Figure 32: Final result of visual rendering with initial boundary – three clusters

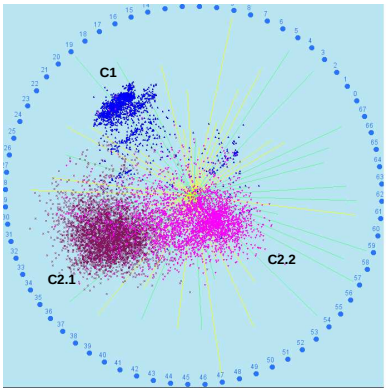

Figure 33: Visualization of ACE clustering result

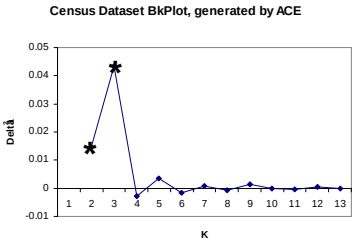

Figure 34: The "Best K" plot for census dataset

In summary, in the above procedure, we show that the combination of the automatic clustering/validation algorithms and the visual rendering method can really help to cross-validate and improve each other. Therefore, cluster analysis under the iVIBRATE framework can provide truly insightful results with much higher confidence level.

8.4 ClusterMap Labeling

Visually clustering the census data under the iVIBRATE framework is a procedure of visual data exploration (Unguided Visual Rendering as defined earlier), in the sense that no initial clustering clues, such as domain knowledge, are provided. For LDS dataset in Section 7.4, we can predefine exact cluster labels with little error and compare the labeling results with the exact cluster labels. However, because of the lack of control labels for the census dataset, we need to change the way of evaluating the labeling algorithms.

As we have initially observed, Adaptive ClusterMap labeling, together with the monitoring tool "ClusterMap Observer", gives much less labeling error and the result becomes very closer to the exact cluster labels with the increase of labeled records. Without the exact control labels for the census dataset, we want to see the

difference between the results of other labeling algorithms and the Adaptive ClusterMap only, which should also be consistent with our interpretation in Section 7.4.

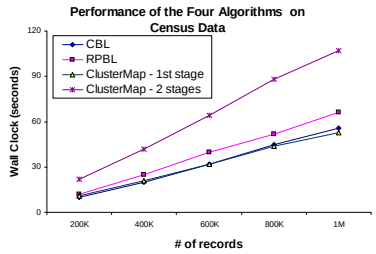

Figure 35: Cost of labeling the census dataset

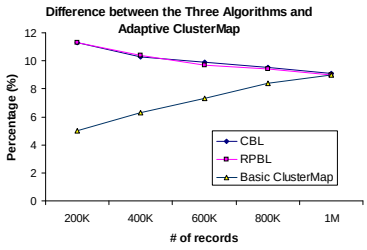

Figure 36: Difference of the labeling results

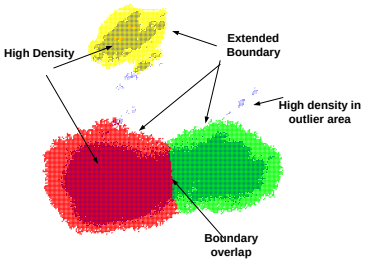

Figure 37: The ClusterMap after labeling 1M records

Figure 35 shows the performance curve of the three labeling algorithms similar to what we observed for the LDS dataset. Figure 36 shows the percentage of difference between the compared algorithms and the adaptive ClusterMap labeling. Since the three clusters are almost regularly shaped and distributed, RPBL and CBL have no big difference as shown in Figure 36. The difference between RPBL/CBL and Adaptive ClusterMap tends to decrease with the increase number of labeled items, due to the automatic boundary extension including more and more points previously regarded as outliers. At the stage a large number of records, for instance, 1 Million records, are labeled, most of the difference should come from outliers. Similarly, the increase of difference between the basic ClusterMap and the adaptive version is solely caused by boundary extension.

Figure 37 demonstrates the cluster and boundary definition after labeling 1 million records. The boundary is evenly extended around the original boundary, therefore, the initial ClusterMap definition is very good. C2 and C3 tend to merge, but C1 is still clearly separated from C2 and C3. Whether to merge C2 and C3 or not may depend on the domain knowledge.

8.5 Exploring the Small Clusters

Some outliers emerge with high density as labeled in Figure 37. Therefore, it might be worthwhile to visualize the outliers separately. Labeling the first million records results in 70632 labeled outliers, about 7% of the total. We sample the outliers again to gain a 10K-record sample set, which is visualized as Figure 38. The initial visualization with the same parameters used in labeling clearly shows that clusters could emerge in the areas A1, A2, A3. By rendering this dataset, we find 5 clusters (the left panel in Figure 38). O1 is mapped to the boundary of the original cluster C2, therefore it should be outliers around C2. O2 and O3 are mapped to the areas A2 and A3 respectively, showing the evidence of new clusters in these areas. O4 and O5 are mapped to the same area A1, which visualizes the delicate structure hiding in A1.

Observing the proximity of the small clusters to the three identified main clusters in Figure 38, we find that O4

and O5 might have close connection with the main cluster C1, and O2 and O3 seem some kind of extension of the main cluster C2.2. Merging them to the main clusters or not should depend on the domain knowledge.

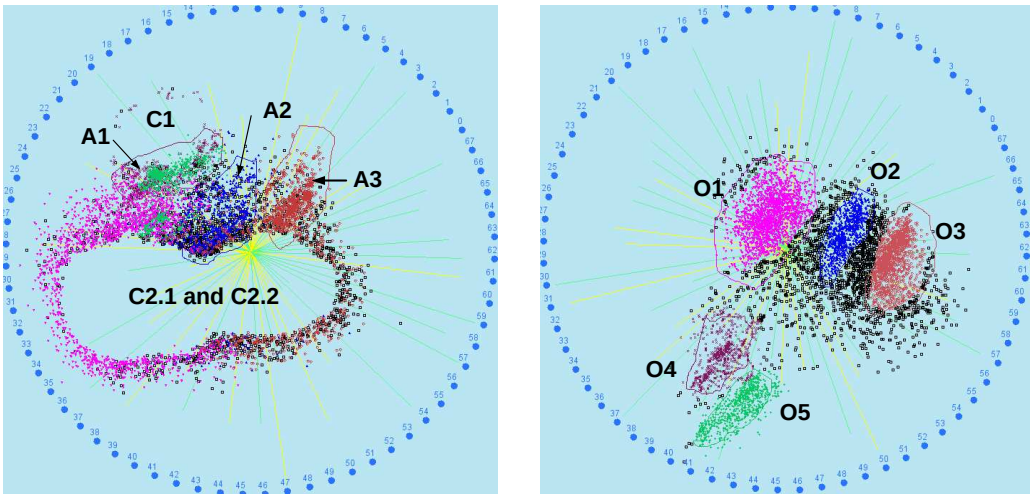

Figure 38: Exploring possible small clusters hiding in outliers

In summary, the adaptive ClusterMap labeling and the small cluster analysis with the census dataset demonstrate the effectiveness and flexibility of iVIBRATE framework in processing very large datasets. With the iVIBRATE framework, irregularly shaped clusters, domain-specific clustering structure, cluster-boundary extension, outlier labeling, and small cluster detection, can be performed and monitored with the help of visualization, which greatly improves the precision of clustering result and increases the confidence about the result.

9 CONCLUSION

We have described iVIBRATE − an interactive-visualization based three-phase framework for clustering large datasets. The VISTA subsystem allows users to interactively view the potential clusters in continuously changing visualizations. More importantly, it can incorporate, validate, and refine the results of any automated clustering algorithms. It is especially effective for dealing with irregularly shaped clusters. To improve the efficiency of visual rendering, we give the heuristic rendering rules and propose a new semi-automated rendering method: automatic random rendering plus automatic dimension-by-dimension rendering for higher dimensional ($>$50D and $<$100D) datasets. The adaptive ClusterMap labeling subsystem preserves the irregular cluster boundary defined in VISTA subsystem, clearly distinguishes the outliers and provides effective mechanisms for fine-tuning and flexible refinement of boundary extensions during the disk labeling phase. We also thoroughly discussed the issues and solutions in integrating the three phases under the iVIBRATE framework, aiming at providing a reliable and stable framework.

Experimental results show that, by incorporating visual cluster rendering and visualization-based disk labeling, the iVIBRATE framework not only greatly improves the quality of intermediate clustering result, but also effectively extends the intermediate result to the entire large dataset with low computational cost and high

precision.

In summary, the iVIBRATE framework presents a very flexible and intuitive approach for exploring clusters in large datasets, which is also an open framework for incorporating existing clustering methods with the power of visualization. We believe that there will be many interesting issues in applying and extending the iVIBRATE framework in the future.

10 ACKNOWLEDGMENTS

This research is partially supported by NSF CNS CCR, NSF ITR, DoE SciDAC, DARPA, CERCS Research Grant, IBM Faculty Award, IBM SUR grant, and HP Equipment Grant.

References

[1] ANKERST, M., BREUNIG, M. M., KRIEGEL, H.-P., AND SANDER, J. OPTICS: Ordering points to identify the clustering structure. *Proc. of ACM SIGMOD Conference* (1999).

[2] ASIMOV. The grand tour. *SIAM Journal of Scientific and Statistical Computing* (1985).

[3] BAEZA-YATES, R., AND RIBEIRO-NETO, B. *Modern Information Retrieval*. Addison Wesley, 1999.

[4] BARBARÁ, D., DUMOUCHEL, W., FALOUTSOS, C., HAAS, P. J., HELLERSTEIN, J. M., IOANNIDIS, Y. E., JAGADISH, H. V., JOHNSON, T., NG, R. T., POOSALA, V., ROSS, K. A., AND SEVCIK, K. C. The new jersey data reduction report. *IEEE Data Eng. Bull. 20*, 4 (1997), 3–45.

[5] BRADLEY, P. S., FAYYAD, U. M., AND REINA, C. Scaling clustering algorithms to large databases. *Proc. of ACM SIGKDD Conference* (1998), 9–15.

[6] CHEN, K., AND LIU, L. Clustermap: Labeling clusters in large datasets via visualization. *Proc. of ACM Conf. on Information and Knowledge Mgt. (CIKM)* (2004).

[7] CHEN, K., AND LIU, L. VISTA: Validating and refining clusters via visualization. *Information Visualization 3*, 4 (2004).

[8] CHEN, K., AND LIU, L. The "best k" for entropy-based categorical clustering. *Proc. of Intl. Conf. on Scientific and Statistical Database Management (SSDBM)* (2005).

[9] CLEVELAND, W. S. Visualizing data. *AT&T Bell Laboratories* (1993).

[10] CONOVER, W. *Practical Nonparametric Statistics*. John Wiley and Sons, 1998.

[11] COOK, D., BUJA, A., CABRERA, J., AND HURLEY, C. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics 23* (1995).

[12] COX, T. F., AND COX, M. A. A. *Multidimensional Scaling.* Chapman&Hall/CRC, 2001.

[13] CUTTING, D. R., KARGER, D. R., PEDERSEN, J. O., AND TUKEY, J. W. Scater/gather: a cluster-based approach to browsing large document collections. *Proc. of ACM SIGIR Conference* (1992).

[14] DHILLON, I. S., MODHA, D. S., AND SPANGLER, W. S. Visualizing class structure of multidimensional data. *the 30th Symposium on the Interface: Computing Science and Statistics* (1998).

[15] DUBES, R. C., AND JAIN, A. K. Validity studies in clustering methodologies. *Pattern Recognition* (1979).

[16] ESTER, M., KRIEGEL, H.-P., SANDER, J., AND XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. *Second International Conference on Knowledge Discovery and Data Mining* (1996).

[17] FALOUTSOS, C., AND LIN, K.-I. D. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *Proc. of ACM SIGMOD Conference* (1995).

[18] FREIDMAN, J. H., BENTLEY, J. L., AND FINKEL, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. on Mathematical Software 3*, 3 (1977).

[19] GALLIER, J. *Geometric Methods and Applications for Computer Science and Engineering.* Springer-Verlag, 2000.

[20] GRAY, J. What is next? a few remaining problems in information technlogy. *Proc. of ACM SIGMOD Conference* (1999).

[21] GUHA, S., RASTOGI, R., AND SHIM, K. CURE: An efficient clustering algorithm for large databases. *Proc. of ACM SIGMOD Conference* (1998).

[22] GUHA, S., RASTOGI, R., AND SHIM, K. ROCK: A robust clustering algorithm for categorical attributes. *Proc. of IEEE Intl. Conf. on Data Eng. (ICDE)* (1999).

[23] HALKIDI, M., BATISTAKIS, Y., AND VAZIRGIANNIS, M. Cluster validity methods: Part I and II. *SIGMOD Record 31*, 2 (2002), 40–45.

[24] HINNEBURG, A., AND KEIM, D. A. An efficient approach to clustering in large multimedia databases with noise. *Proc. of KDD98* (1998).

[25] HINNEBURG, A., KEIM, D. A., AND WAWRYNIUK, M. Visual mining of high-dimensional data. *IEEE Computer Graphics and Applications* (1999).

[26] HOFFMAN, P., GRINSTEIN, G., MARX, K., GROSSE, I., AND STANLEY, E. Dna visual and analytic data mining. *IEEE Visualization* (1997).

[27] HUBER, P. J. Massive data sets workshop: The morning after. *Massive Data Set* (1996).

[28] INSELBERG, A. Multidimensional detective. *IEEE Symposium on Information Visualization* (1997).

[29] IWAYAMA, M., AND TOKUNGA, T. Cluster-based text categorization: a comparison of category search strategies. *Proc. of ACM SIGIR Conference* (1995).

[30] JAIN, A. K., AND DUBES, R. C. *Algorithms for Clustering Data*. Prentice hall, 1988.

[31] JAIN, A. K., AND DUBES, R. C. Data clustering: A review. *ACM Computing Surveys 31* (1999).

[32] KANDOGAN, E. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. *Proc. of ACM SIGKDD Conference* (2001).

[33] KARYPIS, G., HAN, E.-H. S., AND KUMAR, V. Chameleon: hierarchical clustering using dynamic modeling. *IEEE Computer 32*, 8 (1999).

[34] KEIM, D. Visual exploration of large data sets. *ACM Communication 44*, 8 (2001).

[35] LARKIN, J., AND SIMON, H. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science 11* (1987).

[36] LEBLANC, J., WARD, M. O., AND WITTELS, N. Exploring n-dimensional databases. *IEEE Conference on Visualization* (1990).

[37] LI, C., CHANG, E., GARCIA-MOLINA, H., AND WIEDERHOLD, G. Clustering for approximate similarity search in high-dimensional spaces. *IEEE Trans. on Knowledge and Data Eng. 14*, 4 (2002).

[38] LITTLEFIELD, R. J. Using the GLYPH concept to create user-definable display formats. *National Computer Graphics Association* (1983).

[39] LIU, H., AND MOTODA, H. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic Publishers, 1998.

[40] MEEK, C., THIESSON, B., AND HECKERMAN, D. The learning-curve sampling method applied to model-based clustering. *Journal of Machine Learning Research 2* (2002), 397–418.

[41] NEWMAN, M. E. The structure and function of complex networks. *SIAM Review* (2003).

[42] PALMER, C., AND FALOUTSOS, C. Density biased sampling: An improved method for data mining and clustering. *Proc. of ACM SIGMOD Conference* (2000).

[43] RAMASWAMY, L., GEDIK, B., AND LIU, L. A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems (TPDS) 16*, 9 (Sept. 2005).

[44] ROSS, S. M. *Introduction to Probability Models*. Academic Press, 2000.

[45] ROUSSINOV, D., TOLLE, K., RAMSEY, M., AND CHEN, H. Interactive internet search through automatic clustering: an empirical study. *Proc. of ACM SIGIR Conference* (1999).

[46] SCHUTZE, H., AND SILVERSTEIN, C. Projections for efficient document clustering. *Proc. of ACM SIGIR Conference* (1997).

[47] SEO, J., AND SHNEIDERMAN, B. Interactively exploring hierarchical clustering results. *IEEE Computer 35*, 7 (2002).

[48] SHARMA, S. *Applied Multivariate Techniques*. Wiley&Sons, 1995.

[49] SHEIKHOLESLAMI, G., CHATTERJEE, S., AND ZHANG, A. Wavecluster: A multi-resolution clustering approach for very large spatial databases. *Proc. of Very Large Databases Conference (VLDB)* (1998).

[50] SHNEIDERMAN, B. Inventing discovery tools: Combining information visualization with data mining. *Information Visualization 1* (2002).

[51] SILVERSTEIN, C., AND PEDERSEN, J. O. Almost-constant-time clustering of arbitrary corpus subsets. *Proc. of ACM SIGIR Conference* (1997).

[52] SONKA, M., HLAVAC, V., AND BOYLE, R. *Image Processing, Analysis and Machine Vision*. Brooks/Cole Publishing, 1999.

[53] VITTER, J. S. Random sampling with a reservoir. *ACM Trans. on Mathematical Software 11*, 1 (1985).

[54] WARD, M. Xmdvtool: Integrating multiple methods for visualizing multivariate data. *IEEE Visualization* (1994).

[55] WILLETT, P. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management 24*, 5 (1988).

[56] XU, X., ESTER, M., KRIEGEL, H.-P., AND SANDER, J. A distribution-based clustering algorithm for mining in large spatial databases. *Proc. of IEEE Intl. Conf. on Data Eng. (ICDE)* (1998).

[57] YANG, J., WARD, M. O., AND RUNDENSTEINER, E. A. Interactive hierarchical displays: a general framework for visualization and exploration of large multivariate datasets. *Computers and Graphics Journal 27* (2002), 265–283.

[58] YANG, L. Interactive exploration of very large relational datasets through 3d dynamic projections. *Proc. of ACM SIGKDD Conference* (2000).

[59] YANG, Y., AND PEDERSEN, J. O. A comparative study on feature selection in text categorization. *Proceedings of ICML-97, 14th International Conference on Machine Learning* (1997), 412–420.

[60] ZAMIR, O., AND ETZIONI, O. Web document clustering: a feasibility demonstration. *Proc. of ACM SIGIR Conference* (1998).

[61] ZHANG, T., RAMAKRISHNAN, R., AND LIVNY., M. BIRCH: An efficient data clustering method for very large databases. *Proc. of ACM SIGMOD Conference* (1996).