QoS: Quality Driven Data Abstraction Generation For Large Databases *

Charudatta V. Wad, Elke A. Rundensteiner and Matthew O. Ward
Department of Computer Science,
Worcester Polytechnic Institute,
Worcester, MA USA
{charu_w, rundenst, matt}@cs.wpi.edu

ABSTRACT

Data abstraction is the process of reducing a large dataset into one of moderate size, while maintaining dominant characteristics of the original dataset. Data abstraction quality refers to the degree to which the abstraction represents the original data. The quality of an abstraction directly affects the confidence an analyst can have in results derived from such abstracted views about the actual data. Some initial measures to quantify the quality of abstraction have been proposed; however, they currently can only be utilized as an after-thought. An analyst can be made aware of the quality of the data he works with, but he cannot control the quality he desires and the trade-off between the time required to generate the abstraction and its quality. While some analysts require at least a certain minimal level of quality, others must be able to work with certain abstraction quality due to time and resource limitations. To tackle these problems, we propose a new data abstraction generation model, called the QoS model, that presents the performance quality trade-off to the analyst. It then generates an abstraction based on the desired level of quality versus performance as indicated by the analyst. The framework has been integrated into XmdvTool, a freeware multi-variate data visualization tool developed at WPI. Our experimental results show that our approach provides better quality compared to existing abstraction techniques.

1. INTRODUCTION

1.1 Motivation

Data abstraction techniques are commonly used to facilitate the efficient detection of patterns in large datasets and for analyzing a huge database without actually having to explore the original data [1]. Thus, analysts typically infer characteristics of large databases by analyzing the abstracted data rather than looking at the full data. Some abstraction techniques select a subset of the original dataset as its abstraction, such as sampling and filtering, while others construct a new abstract/summary representation, such as clustering and summarizing [1]. Tasks conducted based on abstracted data include pattern detection, cluster analysis, outlier analysis, subspace cluster analysis, filtering and sample analysis [1]. Figures 1(a) and 1(b) represent an example of a dataset and its abstraction. The visualization technique used, called parallel coordinates [13], is a popular multivariate visualization technique. In this method, each dimension corresponds to an axis, and the N axes are organized as uniformly spaced vertical or horizontal lines. A data element in an N-dimensional space manifests itself as a connected set of points, one on each axis. Thus one polyline is generated for representing each data point.

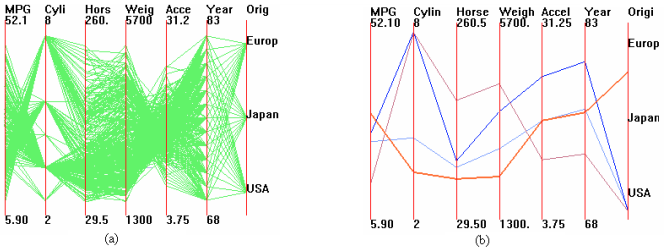

Figure 1: Figure 1(a) Displays the cars dataset using the parallel coordinates visual technique, while Figure 1(b) represents cluster centers of the dataset.

Abstraction quality versus data quality: *Abstraction quality* captures how well the abstracted dataset represents the original dataset. Intuitively, a good data abstraction represents all the main features of the original dataset. Since the abstraction in Figure 1(b) captures all the main clusters present in the original dataset (Figure 1(a)), the abstraction is considered to be of high quality. Lack of knowledge regarding quality can lead to inaccurate results, jeopardizing the reliability of conclusions gleaned from the abstraction. Validating the quality of abstraction is made difficult due to a lack of data abstraction quality measures. Although some initial measures [8] have recently been proposed to measure the data abstraction, those measures do not scale well to higher number of dimensions. Furthermore, a scalable data abstraction measure by itself does not solve the problem. The main problem is a lack of consideration about quality by the abstraction generation process before its commencement.

To further complicate matters, most systems and thus users of these systems assume that the raw data itself is always good. However, real-world data is known to be imperfect, suffering from various forms of defects such as sensor variability, estimation errors, uncertainty, human errors in data entry, and gaps in data gathering. *Data quality* refers to the quality of the underlying data used for

9*This work was supported under NSF grants IIS-0119276 and IIS-00414380.

VLDB '07, September 23-28, 2007, Vienna, Austria.

the abstraction generation. Clearly, if the quality of the underlying data is not considered during abstraction generation, the quality of an abstraction may indeed be adversely affected.

1.2 Existing Abstraction Generation Solutions

Figure 2 sketches the process most commonly used by abstraction generation systems [2] [9].

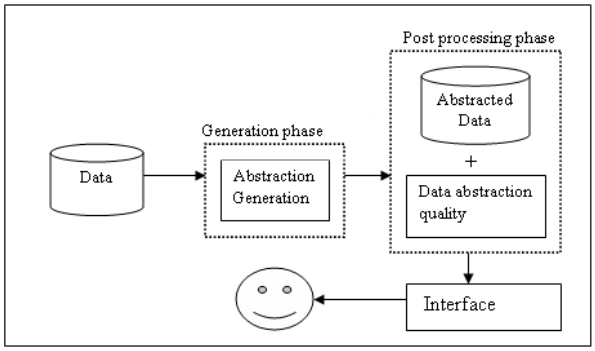

Figure 2: Existing Data Abstraction Solution.

Predicaments of such a process include:

- Quality measures, if available, are plugged in only as an after-thought to calculate the quality of a given abstraction.
- Data abstraction is a one way process. Thus, when an analyst initiates the generation of an abstraction, he cannot indicate the desired level of quality nor control the output of the process in terms of its resulting quality. Rather, he would simply be informed as an afterthought on the quality (or lack thereafter) that has been obtained.
- Furthermore, the analyst doesn't know how much time they should budget for the abstraction process. Without such control, he cannot trade-off the acceptable level of quality with the amount of time he is able to spend on the abstraction process itself.
- Data quality is not taken into account. As discussed earlier, if the data is imperfect (or of low quality), the abstraction result should also reflect the underlying data quality.

1.3 Our Approach

To overcome the above identified problems, we propose to make abstraction generation quality-aware. We present the analyst with a quality-performance trade off indicating the different values of quality measures achievable and time required for the process to generate them. Using these computations, an analyst can demand a quality level beforehand or he can request a certain performance, knowing what quality he can expect and QoS will generate the abstraction accordingly. QoS takes into consideration both the data abstraction quality and underlying data quality to calculate a complete data abstraction quality measure.

The system framework for QoS, depicted in Figure 3, consists of the following main phases:

1. *Pre-processing phase:* We introduce a pre-processing phase to compute the quality-performance trade-off. The computation is done using a multi-dimensional histogram which calculates density information. Two main components in this phase are:

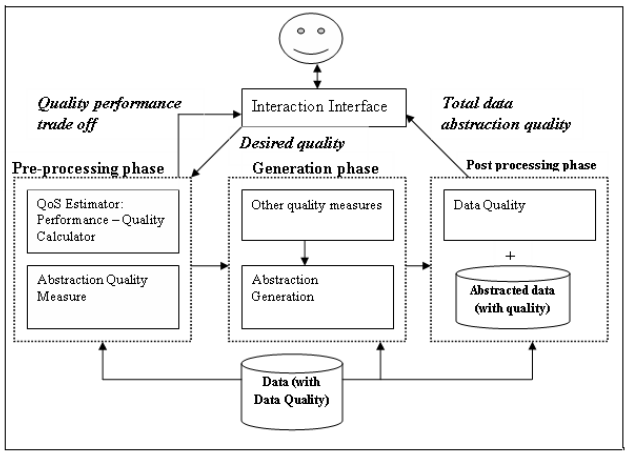

Figure 3: Proposed QoS Framework.

   a) A scalable data abstraction measure to quantify the data abstraction result is proposed, called Multi-dimensional Histogram Difference Measure (MHDM). Other measures [8] could also be plugged in.
   b) The estimator calculates the performance-quality trade-off including confidence intervals and time estimations for the process. This is at the heart of QoS, presenting the analyst with various trade-offs before the process of abstraction commences.

2. *Generation phase:* This process generates an abstraction based on quality values set by the analyst.
3. *Post-processing phase:* It combines the measure of the abstraction with quality of the underlying dataset to determine the overall quality.
4. *Interaction interface:* This interface presents the performance quality trade-off and the final abstraction quality to the analyst.

The QoS framework could be applied to many different data abstraction tasks, including hierarchical sampling, clustering, and selection. However, for the sake of explanation, in the rest of this paper we will focus on clustering for large databases.

2. QOS FOR CLUSTER ANALYSIS

Summarization techniques for data abstraction summarize the data by creating fewer new representatives to convey the underlying data [1]. Clustering is one such technique, where cluster representatives are used to represent the data. Since clustering is memory and computationally intensive, clustering of large databases typically employs sampling as a pre-processing step [2][3].

2.1 Quality Measure (MHDM)

We now propose a measure of abstraction quality for high dimensional data. This multi-dimensional data abstraction quality measure captures the distributions present in a high dimensional dataset. The measure can be calculated before the abstraction is actually generated. The proposed measure, called Multi-dimensional Histogram Difference Measure (MHDM), is a histogram difference method. Histograms are widely used for density and selectivity estimation [4]. MHDM calculates the difference between the

multi-dimensional histogram of the original dataset and that of the abstraction generated from the data. For the measure we assume that the two multi-dimensional histograms (original and abstracted) have the same number of bins, with bin sizes corresponding to the percentage of data falling into that bin. MHDM is the summation of the difference between the corresponding bins. MHDM ranges from 0.0 to 1.0 with 0 implying the worst case MHDM, and 1.0 indicating the best case.

One potential disadvantage of a multi-dimensional histogram is its inability to scale due its high memory requirements [4]. Unfortunately we cannot utilize just 1-dimensional histograms which are less costly, they fail to capture the correlation present in high dimensional data. To overcome the space inefficiency of multi-dimensional histograms [4], we encode the multi-dimensional histogram structure by explicitly associated the multi-dimensional cell address with its cell content value. For an example, Figure 4 represents the formation of an encoded multi-dimensional histogram. For instance, the cell with dimension 1 at bin 5 and dimesnion 2 at bin 2 and dimension 3 at bin 1 having a value of 6 would be encoded explicitly by the pair shown in the figure.

Building the encoded multi-dimensional histogram: Assume the input tuple with *d* dimensions with data values v_1,v_2,..v_d.

Step I: We partition each of the d dimensions into a number of distinct partitions. For simplicity, we'll assume here that there are exactly n such partitions for each dimension, though other more sophisticated strategies could be employed for bin sizing in the future.

The partitioning of the dimension i is denoted as u_1^i, u_2^i, ... u_n^i with n the number of partitions. For each input tuple v_1,v_2,..v_d, we determine which bin b of dimension i its i^{th} value v_i falls into. Given that each tuple value is mapped to a particular partition, we have *d* partition numbers for a given input tuple. Let us denote this by u_{i1}^1,u_{i2}^2,..u_{id}^d, with i_j the partition number for the dimensions. Thus, the number of 1-dimensional partitions formed directly influence the number of multi-dimensional bins formed.

Step II: We encode the multi-dimensional bin from partition numbers obtained from each dimension by appending the bin numbers into one code, $u_{i1}^1 u_{i2}^2..u_{id}^d$ is the multi-dimensional bin corresponding to the example input tuple above. Thus, if most of the d-dimensional cells remain empty, our histogram is relatively small. Most real datasets are very sparse in nature (confirmed by our experimental study in Section 4). Thus this technique saves a lot of memory in practice.

Advantages of this explicit encoding of a full matrix representation approach include :

- We never encode empty bins, leading to huge savings in terms of memory in practice.
- The algorithm has a linear complexity (in the number of data points), and thus can build multidimensional histograms efficiently even for high dimensional data.

MDHM can be expressed by the following equation:

$$MHDM = 1.0 - \frac{\sum_{i=1}^{N} |Po_i - Ps_i|}{MAX_{Ph}} \quad (1)$$

- Po_i is the percentage of data that falls into the i-th bin of the original histogram;

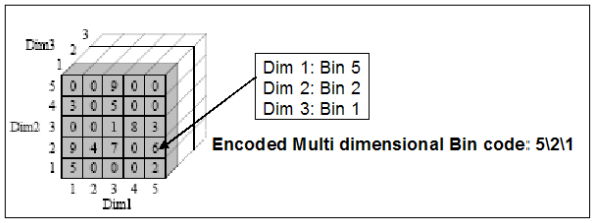

Figure 4: Formation of encoded multi-dimensional histogram.

- Ps_i is the percentage of data that fall into the i-th bin of the abstracted histogram;
- MAX_{Ph} is the maximum histogram difference.

Need for MHDM: In clustering of large databases, if the samples used for clustering are chosen randomly, they fail to represent the original dataset. In that case, the clustering process fails to abstract the original dataset. It is independent of the clustering algorithm used and quality of clusters formed. There is no intuitive way of setting the "correct" sampling rate for a dataset. In absence of measures to guide the process, the easiest method to ensure high data abstraction quality is to increase the sampling rate. The user may over-sample the database yielding a poor clustering performance without guaranteeing necessarily improvement in quality. Also, users might under-sample the dataset leading to low data abstraction quality. In that case, the clustering result might not be accurate. Thus, misleading the users with clustering results that may not represent the original dataset. Thus, even though sampling can be a direct representation of quality, setting the correct sampling rate requires a quality measure.

2.1.1 Noise Elimination

Real world data is often fraught with noise. Clearly, noise elimination is crucial for high quality abstractions. The multi-dimensional histogram of the original data is thus regulated to filter noise. Here we propose one method in particular that is targeting the elimination of noise in support of the task of clustering; however, other methods for the elimination of noise may need to be designed to support alternate tasks. Clearly, noise in the context of clustering may be important information when in search of outliers.

Our proposed cluster-centric noise elimination phase consists of eliminating bins whose bin count is below a threshold (γ). This threshold γ can either be empirically determined (explained in Section 4) or set by the analyst. Intuitively, we observe that the bin count of a multi-dimensional bin will be below a threshold if either the point is a random noise or the point belongs to the edge of a cluster.

Figure 5 displays a grid representing a 2-dimensional histogram placed over the data. Ignoring points from low bin counts may have the side effect of ignoring points from the edge of the clusters. However, since we are interested in picking more points from near the center of the cluster rather than its edges, ignoring points from the edges effectively adds more weight to the points in the center. This improves the abstraction quality, as our experimental study confirms (see Section 4). It also decreases the number of multi-dimensional bins to be maintained, increasing the efficiency of the QoS estimator (Section 2.2).

2.2 QoS Estimator

The QoS estimator computes the performance quality trade-off by generating a look-up table that indicates the relationship be-

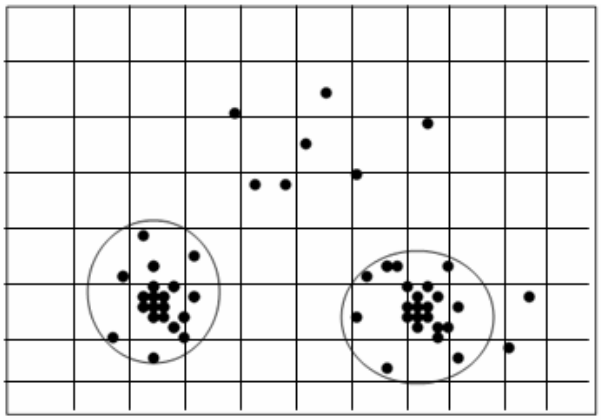

Figure 5: Existence of noise in datasets.

tween MHDM, the sampling level and the estimated time required for clustering. Figure 6 shows an example of a look-up table generated by the QoS estimator. Since sampling is the preliminary step for clustering, the abstraction quality largely depends on the sampling. If the samples chosen for clustering do not represent the original dataset well, the abstraction quality of the clusters can be low. Thus, the abstraction quality is determined by sampling. Various sampling techniques are defined in the literature which can be used in this framework. Palmer et al. devised the strategy of density biased sampling [5], a probability based approach, which samples more from a dense region and less from a sparse region.

According to density biased sampling [5]: suppose that we have n values x_1, x_2, ... x_n that are partitioned into g groups that have sizes n_1, n_2,, n_g and we want to generate a sample with the expected size M in which the probability of point x_i is dependent on the size of the group containing x_i.

To bias the sample size, the probability function is defined as [5]:

$$f(n_i) = \frac{\beta}{n_i^e} \tag{2}$$

where n_i is the number of points in group g_i and e is a constant. The number of points selected from group g_i:

$$n = f(n_i) * n_i \tag{3}$$

β is defined based on the sample size (M) as follows:

$$\beta = \frac{M}{\sum_{i=1}^{g} n_i^{1-e}} \tag{4}$$

Group formation: It is very important to form groups based on density for density biased sampling [5] to be effective.The group assignment is done using the encoded multi-dimensional histogram. Each bin is treated as group of points used by density biased sampling.

Algorithm for estimation using density biased sampling: A look-up table is generated after the multi-dimensional histogram for the original data has been formed. Starting with sampling level α, the number of points falling in each bin are calculated. This enables us to calculate MHDM for sampling level α. It is repeated until MHDM reaches the maximum value of 1.0.

The look-up table (as shown in Figure 6) will have a sampling level, minimum quality level followed for the sampling, and the

Algorithm 1 Populating look up table

*Input: x= Initial sampling rate, and α = Increment in the sampling rate. /*Populating $lookup\_table$ for performance-quality trade off. Initialize by setting M $\leftarrow$ x, and calculating β from Equation 4. */*

91: **while** (*MHDM* $\leq$ **1**) **do**
92: **for** each $bin \in multi - dimensionalhistogram$ **do**
93: Number of points selected from each group from equation 3;
94: **end for**
95: Compute MHDM for M ;
96: Compute time and confidence interval;
97: Update *lookup_table with sampling rate and MHDM*;
98: *M* $\leftarrow$ *M*+α, compute β ;
99: **end while**

Output: lookup table of performance quality trade off.

MHDM	Sampling level (%)	Time(sec)
.50	0.1	2.38
.60	0.5	4.99
.70	1.2	7.56
.80	2.7	10.16
.90	3	12.33
1.0	3.8	15.3

Figure 6: Sample look-up table created by QoS estimator.

time required for the process to complete. Whenever an analyst chooses a quality value, the value closest to it is returned.

2.3 Interaction Interface

The Interaction module allows the analyst to attain information on the quality performance trade off. The analyst can set one of three values: data abstraction quality (MHDM) value, sampling rate, and time for completion of the clustering process.

2.4 Abstraction Generator

Once the analyst decides on the quality and other performance settings he desires, the interaction interface passes the sampling level to the abstraction generator. The abstraction generator samples the database using density biased sampling with a sampling level set by the interaction interface. The abstraction generator then passes the generated samples to a clustering algorithm. At this point, we can use any existing clustering technique [1] to cluster the data.

2.5 Inclusion of Data Quality

As a last step, the quality of the underlying data is incorporated into the abstraction result. As is commonly done [12] [14], we assume that each data tuple has an associated record quality. Every cluster consists of data points of the original dataset. Thus, to calculate the total abstraction quality, we incorporate the data quality of all its members using some statistical function. Many alternative methods are possible, such as arithmetic mean and standard deviation, median values, geometric mean, root mean square and so on. For illustration purposes, henceforth, we chose to represent the data quality of clusters using the arithmetic mean of the record qualities,

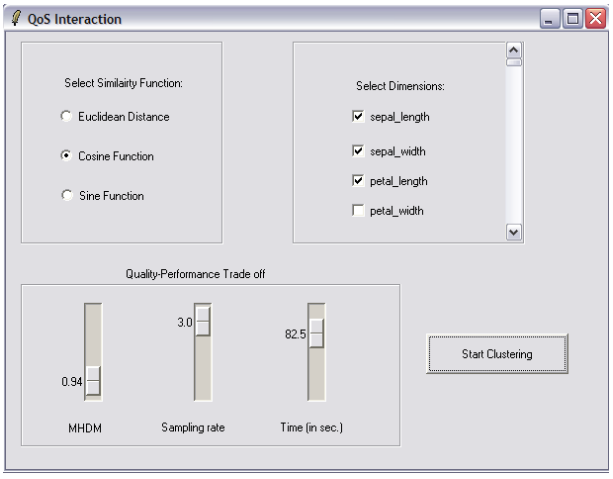

Figure 7: QoS sampling interface.

called Cluster Data Quality (CDQ).

The Cluster Data Quality (CDQ) can be expressed as:

$$CDQ = \frac{\sum_{i=1}^{n} RecordQuality}{n} \quad (5)$$

- CDQ: Cluster Data Quality
- Record Quality: Data quality of the record [0 : 1].
- n: number of points in the cluster.

2.6 Total Abstraction Quality

The MHDM of the data clustered is identical to the value set by the analyst in the pre-processing phase. However, we can also evaluate the performance of the clustering algorithm using a quality measure [9]. One possible clustering quality measure can be the average distance of every point from its nearest cluster center [9]. We can plug these clustering quality measures in the generation phase to find the quality of clustering performed, which we call Cluster Quality (CQ). MHDM can be visualized as a global measure on the entire dataset, whereas clustering quality measure gives a quality value for each cluster formed.

Thus, the total data abstraction quality can be calculated as the weighted average of cluster data quality, cluster quality and abstraction quality:

$$TAQ = \frac{\rho * CQ + \delta * CDQ + \lambda * MHDM}{3} \quad (6)$$

- TAQ: Total data abstraction quality;
- ρ: Weight associated with clustering quality;
- CQ: Cluster quality;
- δ: Weight associated with the data quality;
- CDQ: Cluster data quality;
- MHDM: Abstraction quality;
- λ: Weight associated with abstraction quality.

ρ, δ and λ can be user set parameters or can be set to 1 to default to the arithmetic mean.

We display the TAQ visually using an InterRing display [15].

3. RELATED WORK

Recently some abstraction measures are introduced in the field of information visualization. Cui et. al. [8] proposed a histogram based measure. In contrast to HDM [8], our measure uses a multi-dimensional histogram to capture co-relations in higher dimensional data. Sampling and clustering has been extensively studied [1] [2] [3]. Olken et. al [11] proposed the idea of random sampling for data analysis which was improved by C. Palmer et. al [5] using density biased sampling. We employ density biased sampling as a sampling method in QoS due to its density preservation property.

Human interaction in the field of clustering was advocated by K. Chen et al. [6] via Vista Software. Vista allows manual clustering of databases by analysts. Widom proposed a data model called Trio [14] which incorporates lineage and accuracy of the data. However, Trio does not deal with quality of abstracted data nor with clustering tasks – rather, it assumes simple sql-style queries are being processed against the data. In other words, the proposed data model is in effect an extended relational model.

4. EXPERIMENTAL EVALUATION

We have evaluated the framework using both real and synthetic datasets. The framework is integrated into XmdvTool, a public domain data visualization tool [7] developed at WPI. Experiments were conducted on Pentium 4 (1.66 GHz) running on Microsoft Windows XP with 1.0 GB RAM. We have conducted experiments for assessing the different components of QoS.

4.1 Practicality of Encoded Multi-Dimensional Histograms for Real Datasets

In this experiment, we formed encoded multi-dimensional histograms for numerous real dimensional datasets with different numbers of dimensions, such as Iris, Out5d, Cars, Aaup, Census_income and Supercos2.

Figure 8 displays the comparisons of the number of bins actually formed and the maximum number of bins possible. As seen from Figure 8, savings (difference between maximum possible bins and bins actually formed) increase enormously. This confirms the fact that the real datasets are sparse in nature and our encoding based approach indeed saves memory in practice.

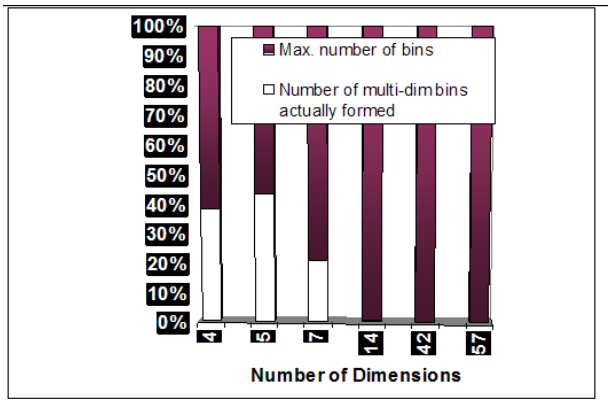

Figure 8: Savings for real datasets.

The savings also increase greatly if we form a larger number of partitions. Figure 9 represents the effect of increasing the number of partitions on number of multi-dimensional bins formed and thus on the MHDM.

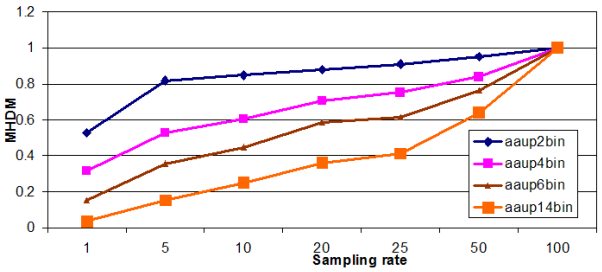

Figure 9: Effect of increasing number of 1-d partitions for Aaup dataset.

4.2 Validating QoS Clustering Accuracy

For this experiment, we used synthetic datasets generated with a known number of clusters. We compare the clustering result of the dataset without sampling and after applying QoS. We used a K-means algorithm to find the RMS error between the cluster centers of the original datasets and those of the abstractions generated by QoS. RMS error (ϵ) can be defined as:

$$\epsilon = \sqrt{\frac{\sum_{i=1}^{m}(c_o^i - c_s^i)^2}{m}} \quad (7)$$

- c_o^i: original cluster center;
- c_s^i: cluster center of the abstraction;
- m: number of clusters.

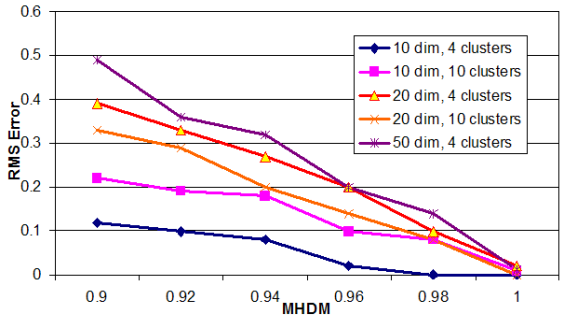

Figure 10: Decrease in RMS error with increase in MHDM.

As seen from Figure 10, the RMS error decreases with an increase in MHDM. Thus, an increase in MHDM leads to a more accurate identification of clusters.

4.3 Conformance of MHDM with Average Distortion

For this experiment, we calculate the average distortion of the dataset. Average distortion is the average distance of a datapoint from its cluster center. Figure 11 represents the average distortion with and without noise elimination phase. It can be noticed that in absence of noise elimination phase, with an increase in MHDM (and thus, the chosen sampling level) the average distortion increases. This is because of the introduction of noise in sampling. Thus, we introduced a noise elimination phase, γ was set at 0.2 percent, eliminating all the bins below the threshold. With the introduction of the noise elimination phase, the average distortion decreases linearly with the increase in MHDM. Since, K-means clustering algorithm was used, the number of clusters formed remains the same. In short, the noise elimination stage facilitates formation of dense clusters.

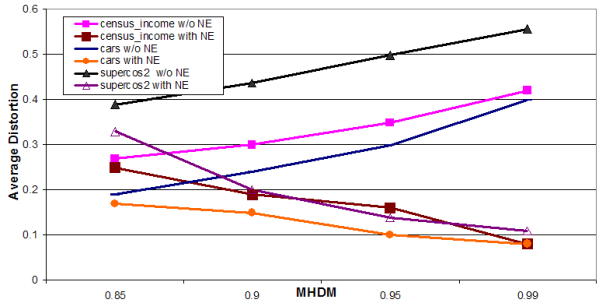

Figure 11: Average distortion with and without noise elimination for various real datasets.

5. CONCLUSIONS

To enable effective use of abstraction results, we observe that one, data abstraction quality must be quantified and two that the analyst must be given the ability to control this quality. We address this problem by making the abstraction process both quality-aware and interactive. In particular, we have introduced an abstraction generation framework which enables the analyst to trade-off between the time dedicated for the generation of the abstraction versus the level of quality one can expect to achieve as result of the abstraction process. This framework has been successfully implemented, and then incorporated into the XmdvTool data visualization application. Experiments have demonstrated that our proposed framework indeed improves the quality and scalability of the abstraction process to a great extent. The framework is general in that it can be used for any application working with abstraction generation in support of tasks such as clustering.

6. REFERENCES

[1] P. Berkhin. Survey of Clustering Data Mining Techniques. Technical report, Accrue Software, Inc., San Jose, CA, 2002.

[2] T. Zhang, R. Ramakrishnan, M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 103-114, Montreal, Canada, June 1996.

[3] S. Guha, R. Rastogi, K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proceedings of ACM SIGMOD conf.* Washington, USA,1998.

[4] H. Wang, K. Sevcik. A Multi-dimensional Histogram for Selectivity Estimation and Fast Approximate Query Answering. In *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*, Pages: 328 - 342, Toronto, Ontario, Canada, 2003.

[5] C. Palmer, C. Faloutsos. Density Biased Sampling: An Improved Method for Data Mining and Clustering. In *Proceedings of ACM SIGMOD*, May 2000.

[6] K. Chen, L. Liu. VISTA: Validating and Refining Clusters via Visualization. In *Proceedings of the 3rd IEEE International Conf. on Data Mining*, Page: 501, 2003.

[7] M. Ward. XmdvTool: Integrating Multiple methods of Visualizing Multivariate Data. In *In Proc. Visualization*, pages 326331,1996. .

[8] Q. Cui, M. Ward, E. Rundensteiner, J. Yang. Measuring Data Abstraction Quality in Multiresolution Visualization. In

IEEE Symposium on Information Visualization (InfoVis 2006), October 2006.

[9] F. Kovcs, C. Legny, A. Babos. Cluster Validity Measurement Techniques. In *6th International Symposium of Hungarian Researchers on Computational Intelligence*, Budapest, Nov. 2005.

[10] S. Thompson. Sampling. *John Wiley and Sons, Inc., New York, 2nd Edition*, 1992.

[11] F. Olken, D. Rotem. Random Sampling from Database Files: A Survey. *Proc. Fifth Int'l Conf. Statistical and Scientific Database Management*, 1990.

[12] Z. Xie, S. Huang, M. Ward, E. Rundensteiner. Exploratory Visualization of Multivariate Data with Variable Quality. In *IEEE Symposium on Visual Analytics Science and Technology*, pp183-190, October 2006.

[13] E. Wegman. Hyperdimensional data analysis using parallel coordinates. In *Journal of the American Statistical Association*,411(85):664675, 1990.

[14] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR '05)*, Pacific Grove, California, January 2005.

[15] J. Yang, M. Ward and E. Rundensteiner. InterRing: An Interactive Tool for Visually Navigating and Manipulating Hierarchical Structures. In *Proc. IEEE Symposium on Information Visualization*, : 77-84, 2002.