Projective Clustering by Histograms

Eric Ka Ka Ng, Ada Wai-chee Fu and Raymond Chi-Wing Wong, *Member, IEEE*

***Abstract*— Recent research suggests that clustering for high dimensional data should involve searching for ”hidden” subspaces with lower dimensionalities, in which patterns can be observed when data objects are projected onto the subspaces. Discovering such inter-attribute correlations and location of the corresponding clusters is known as the projective clustering problem. In this paper, we propose an efficient projective clustering technique by histogram construction (EPCH). The histograms help to generate “signatures”, where a signature corresponds to some region in some subspace, and signatures with a large number of data objects are identified as the regions for subspace clusters. Hence, projected clusters and their corresponding subspaces can be uncovered. Compared to the best previous methods to our knowledge, this approach is more flexible in that less prior knowledge on the data set is required, and it is also much more efficient. Our experiments compare behaviors and performances of this approach and other projective clustering algorithms with different data characteristics. The results show that our technique is scalable to very large databases, and it is able to return accurate clustering result.**

***Index Terms*— Projective clustering, histogram, subspace**

I. Introduction

Clustering is being applied to many practical problems such as image segmentation, pattern recognition, trend analysis, etc. It is often considered an important method in data mining. Here, we state the problem of traditional clustering as in [13] : *Given a number of objects, each of which is described by a set of numerical measures, devise a scheme for dividing the objects into a number of groups such that objects within the same group are similar in some respect and unlike those from other groups. The number of groups and the characteristics of each group are to be determined.*

Surveys on traditional clustering techniques and concepts can be found in [16]. Traditional clustering algorithms are different in their terminologies, cluster representations, assumptions for the components of the clustering process and the contexts in which clustering is used. They are often classified as *hierarchical clustering, partitional clustering, optimization techniques and density search techniques*. However, most of them are originally aimed for tackling clustering problems with low dimensional data.

With high dimensional data commonly found nowadays (e.g. typical relational database contains tens up to hundreds of attributes), we face the “dimensionality curse” problem, and most traditional clustering algorithms cannot produce satisfactory results. Recent theoretical results [17] show that in high dimensional space (e.g. 10 - 15), distances between every pair of data objects are almost the same for a wide variety of data distributions and distance functions. The concept of proximity and neighborhood can hardly be applied in such high dimensional space, and thus natural cluster would not exist in the full dimensional space.

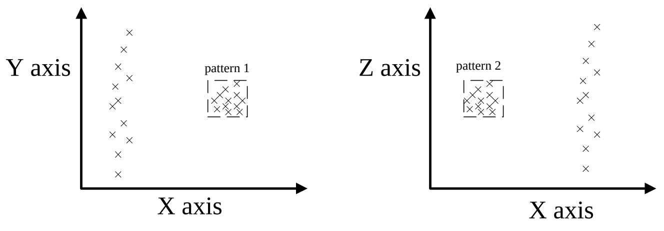

Fig. 1. Projected clusters on subspace XY and XZ.

The traditional approach to the clustering problem is not able to meet the challenges in high dimensional space, where patterns typically exist in small subsets of attributes. Feature selection techniques or methods such as Principal Component Analysis (PCA) are proposed to reduce the dimensionality of the data, by projecting all the data objects onto a subspace while minimizing the information loss. However, in real life applications, correlations among dimensions may often be localized to different clusters. Different patterns can only be uncovered when we consider projections of the data objects onto different subspaces. In such cases, any attempt to reduce the dimensionality of the whole database would bring substantial information loss.

Consider Figure 1 from [4], traditional clustering algorithm would fail to uncover any cluster in the 3-dimensional space **XYZ**. On the other hand, if we do a feature selection or reduce the whole database to space **XY** or space **XZ** only, one of the patterns would be missed, since each dimension is relevant to at least one of the patterns. In this scenario, it is proposed that “projected clusters” [4] would give us more meaningful information about the underlying clustering structure. Projected cluster is defined in [5] as the following:

*A **projected cluster** is a set ε of orthogonal vectors together with a set $\mathcal{C}$ of data points such that the points in $\mathcal{C}$ are closely clustered in the subspace defined by the vectors ε. The subspace defined by the vectors in ε may have much lower dimensionality than in the full dimensional space.*

Projective clustering algorithms such as PROCLUS in [4], and ORCLUS in [5], [3] have been shown to give good quality result, and continue to attract new ideas, such as those in [21]. PROCLUS and ORCLUS aim at discovering projected clusters with different properties. PROCLUS discovers groups of data objects located closely in each of the related dimensions in its associated subspace. In such case, the data objects would be spread along certain directions parallel to the original axes. ORCLUS solves a more general problem. It aims to detect arbitrarily oriented subspaces formed by any set of orthogonal vectors. The set of orthogonal vectors need not be parallel to the original axes. For example, for a data set in the 3-d space **ABC**, there may exist a projected cluster with 2-d subspace,

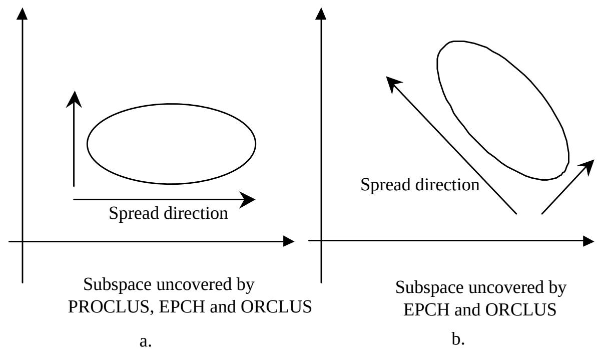
Spread direction
Subspace uncovered by PROCLUS, EPCH and ORCLUS
a.
Spread direction
Subspace uncovered by EPCH and ORCLUS
b.

Fig. 2. Subspaces detected by different algorithms

composed by 2 orthogonal vectors $v_1 = 0.5A + 0.2B + 0.4C$ and $v_2 = 0.2A + 0.5B - 0.5C$. Neither v_1 nor v_2 is parallel to the original axes.

As pointed out in [21], in real life applications, often only the original coordinate axes are meaningful and are interpretable. For example, suppose a customer database contains many attributes, and three of the attributes are: age, income and number of family members. We may discover one projected cluster in the subspace [age, income], and another projected cluster in the subspace [age, number of dependents], with the following interpretation: a typical group of customers with ages in range A, incomes with range B and independent of the number of dependents is interested in the new product; an interesting correlation exists between ages in range C and the number of dependents in range D. Such interpretations rely on the meanings of the original attributes (dimensionalities). Hence the problem definition of PROCLUS is more desirable in some cases, and this is the focus of a recent work [21], in which a Monte Carlo algorithm is proposed.

Both methods of PROCLUS and ORCLUS require the users to provide the number of clusters *k*, and the dimensionality *l* of the subspaces. The clustering quality could be greatly affected if the user cannot estimate the correct values. ORCLUS restricts that the dimensionality of each uncovered subspaces need to be the same. However, associated subspaces with different dimensionalities in the same data set are more realistic. For example, some patterns exist in the 2-d subspace [age, income], while other patterns may exist in 1-d subspace [age].

A. Our Contributions.

The objective of our proposed method, which we call **EPCH** (Efficient Projective Clustering by Histograms), is focused on *uncovering projected clusters with varying dimensionality, without requiring the users to input the average dimensionality of associated subspaces, and the number of clusters that naturally exist in the data set*. Figure 2 illustrates the subspaces with different properties that can be detected by the three methods.

EPCH requires very little prior knowledge about the data. Unlike PROCLUS and ORCLUS, the user of our algorithm does not need to provide the dimensionality of the subspaces and the number of natural clusters. A general user need to provide only one input, *max_no_cluster*. It represents the maximum number of clusters the user is interested to uncover. In case the number of natural clusters is smaller than *max_no_cluster*, it will return all the discovered clusters (there may be less than *max_no_cluster* such clusters). In other cases, it will return the top *max_no_cluster* ranked clusters. Therefore, an inaccurate estimation of this parameter will not affect the accuracy of the clustering output. There are some tuning parameters with default values which can be tuned: (1) an upper bound f of the spread of a cluster in each projection domain. (2) a value k to speed up the final clustering process. A bigger value of k can be set with more system resources. The idea is that if a user wants at most m clusters, then km possible candidates will be considered at the last stage. (3) Either a threshold of the cluster membership degree for outlier/noise determination or the percentage of data expected to be outlier/noise. This is optional, we may also choose not to return any outlier/noise.

Our method can report the regions (hyper-rectangles) where the clusters are located in its associated subspace. We believe that this information is very useful to most users. For example, reporting that the cluster with subspace [age, income] is located in $25 < age < 28 \bigwedge 20000 < income < 25000$ can give the user a good idea on the cluster pattern.

To summarize, our proposed algorithm is efficient and requires less parameters. Our experiments show that it generates high quality results. In addition, our method can easily handle clusters of varying densities and/or varying dimensionalities. Our other contribution is to propose a new measure for the clustering result quality, known as the “coverage ratio”, which can complement the inadequacy of previous measures.

II. Related Works

Previous methods such as DBSCAN [12], OPTICS [9], BIRCH [26], and STING [25], give promising results for low dimensional data. However, they do not aim at clustering high-dimensional data. They become computationally expensive and also ineffective with growing dimensionality. As pointed out by [14], the curse of dimensionality has a severe impact on their resulting clustering quality, and continues to pose a challenge to clustering algorithms at a fundamental level.

The automatical discovery of interesting subspace is first studied as the subspace clustering problem in CLIQUE [6]. CLIQUE targets at subspaces defined by axis-parallel domains. CLIQUE has an inherit problem in that there is only one density threshold for subspaces of all dimensionalities, which is not justifiable since the sparseness of projected data naturally increases with the number of dimensions in the projected subspace. Following that, ENCLUS [10] extends the idea of subspace clustering by using entropy to further prune away uninteresting subspaces. These approaches report ”dense” regions in each discovered interesting subspace.

Following that, PROCLUS and ORCLUS are designed to uncover projected clusters and their associated subspaces. PROCLUS is targeted to find clusters in subspaces formed by axis-parallel vectors, while ORCLUS computes clusters with arbitrarily oriented subspaces. We aim to solve the similar problem as PROCLUS and ORCLUS, with higher efficiency, accuracy and more flexibility. Here we give more descriptions about the ideas of PROCLUS and ORLCUS.

PROCLUS: PROCLUS employs *Manhattan Segmental Distance* as similarity measurement to quantitatively describe how good a projected cluster is. Specifically, for any two d-dimensional data points $x_1 = (x_{1,1}, \ldots, x_{1,d})$ and $x_2 = (x_{2,1}, \ldots, x_{2,d})$, and for any subset S of the set of dimensions, $|S| \leq d$, the Manhattan segmental distance between x_1 and x_2 relative to S is given by $d_S(x_1, x_2) = \frac{(\sum_{i \in S} |x_{1,i} - x_{2,i}|)}{|S|}$. Note that this distance metric tries not to penalize higher dimensionalities by a normalization with the number of dimensions. With this distance metric, PROCLUS tries to minimize the intra-cluster distance, which is the sum of distances between data objects in a cluster and the centroid of the cluster. It works like a k–Means algorithm, extended with the idea of projected clustering. First, a greedy approach is applied to select a set of potential medoids (the centers of the clusters). With the set of medoids, it estimates the correlated subspace for each cluster by examining its *locality*. Locality is defined as the set of data objects in a neighborhood region in the full dimensional space. The projections of these data objects on different single dimensions are examined to find those dimensions that are having closer average distances to the corresponding medoids. These are chosen as the dimensions of the correlated subspace. After the estimation of subspaces, data objects are assigned to its closest medoid with the distance measured with respect to the corresponding subspace. The quality of clustering, which is the sum of intra-cluster distance, is evaluated. Medoids are replaced in a hill-climbing approach, which targets to give an improvement on the clustering quality.

One problem we can see is that the full dimensionality is used in forming the locality. This may not include the real neighbors in the correlated subspace and may include unrelated points in terms of the subspace. In fact, according to [17] it makes little sense to look for neighbors in the high-dimensional space. Also, the parameters of k and l, the dimensionality of clusters, may greatly affect the result quality.

ORCLUS: ORCLUS makes use of Singular Value Decomposition (SVD), which is a well known technique for dimension reduction, with a least loss of information. SVD transforms the data to a new coordinate system (defined by a set of eigenvectors) in which the correlations in the data are minimized. In contrast, ORCLUS chooses the eigenvectors with minimum spread (eigenvalue) to do the projection, so that the greatest amount of *similarity* among the data points in the clusters can be detected. ORCLUS works as a hierarchical merging method, starting with a group of initial seeds. During the merging process, the dimensionalities of the subspaces associated with each clusters would be gradually reduced, by using the SVD technique mentioned above. The merging would be terminated when the number of clusters and the dimensionalities of subspaces reach the user input parameters.

The running time of ORCLUS is expected to be much longer than PROCLUS. In its merging stage, the eigenvectors with the least spread for each pair of the remaining clusters have to be computed, which takes $O(D^3)$ time by using ECF (Extended Cluster Feature Factor), where D is the dimensionality of the original space. This can be prohibitively expensive for very large database and high dimensionality. Another limitation of the method is that the dimensionalities of all clusters are assumed to be the same and also assumed to be known beforehand. These assumptions are likely not true in real cases and from our experiments when the assumptions are not true, the performance can deteriorate greatly.

[2] proposes a human computer cooperative system for high dimensional clustering. There are three iterative steps: (1) determine subspaces with well polarized data, this is by means of principal component analysis. The step will output 2-dimensional projections for the next step. (2) user interaction to visually separate the clusters in each 2-dimensional projection from the previous step. (3) store user interactions in the form of IdStrings, which will lead to the resulting clusters. In Step (1), potential subspaces (polarized subspaces) are identified. It is an iterative process, starting with full subspace and reducing the dimensionality ”gradually” until it reaches 2. In each iteration, there is a set of k random enchor points. For each anchor point p, a set of neighboring points M in the current subspace is located, and this set of data is subjected to principal component analysis to determine a subset of dimensions where M are tightly located around p. (In addition to k, there is a user parameter s for the number of points in M.) However, similar to PROCLUS, the initial neighborhood is determined in the full space. According to [17], in general it makes little sense to look for neighbors in high-dimensional space. Therefore, the basic step of the method is in doubt.

Recently, [21] proposes a Monte Carlo algorithm to approximate an ”optimal projective cluster”. Specifically, the quality of a projected cluster is defined as $\mu(|\mathcal{C}|, |\mathcal{D}|)$, where $|\mathcal{C}|$ is the number of data objects, $|\mathcal{D}|$ is the dimensionality of the subspace, and μ is a monotonically increasing function in each argument. Intuitively, a ”good” cluster should contain as many points as possible, and associated with high dimensionality. However, these two objectives are at odds. A β-balanced measure is used to balance the trade-off between these two objectives. [21] is targeted for axis-parallel clusters as in PROCLUS.

III. Projected Clustering using Histogram

Given N data objects, with dimensionality D, let $max\_no\_cluster$ be the maximum number of clusters of interest to the user. Assume that d-dimensional histograms are constructed.

Definition 1: A *data object* x_i is a single data item. It is represented by a vector of D numerical measurements in the D-dimensional space : $x_i = (x_{i,1}, ..., x_{i,D})$, where $x_{i,j}$ is an **attribute** of x_i.

Definition 2: A **subspace** S_i with dimensionality d_i is defined by a set of d_i orthogonal vectors, where $d_i < D$.

Definition 3: A **projected cluster**, or simply a cluster, C_i associated with subspace S_i is a set of data objects, which are closely clustered when projected in the subspace S_i.

Definition 4: We call the subspace of a projected cluster the **associated subspace** of the cluster. We call the dimensions being included in the associated subspace the **bounded dimensions** of the cluster, and others as **unbounded dimensions**. Similarly, we call any subset of the associated subspace **bounded subspace**. We would use *A*, *B*, *C* etc. to denote dimension, and *AB*, *ABC* etc. to denote subspace.

A. Histograms

During the data summarization process, a **histogram** is widely used, which generates an approximation, $\hat{f}(x)$, to the original measurement of a variable x, $f(x)$, especially when the computational efficiency is concerned. Let us first consider the 1-dimensional case, we divide each dimension into a set of equisized **bins** with a **binwidth** h.[1] The **histogram** is defined by

$$\hat{f}(x) = \frac{1}{Nh}(\text{number of objects in the same bin as } x)$$

One important parameter of a histogram is the bin width h, as it controls the trade-off between undersmoothing or oversmoothing the true distribution. Sturges' rule [24] suggests that if the data follows a normal distribution, the number of *equisized bins* for an ideal frequency histogram should be $1+log_2N$. For the d-dimensional histograms we suggest to use $(1 + log_2N)^d$ bins. In our implementation, we use a greater number for the 1-d histograms since the computational cost is still very low. We use at least $(1 + log_2N)^2$ bins for 2-d histograms. Note that Sturges' rule targets at preserving the shape of the underlying distribution. Our objective is humbler, we only want to uncover any dense region, there is no need to preserve the shape of the distribution. Therefore, we could use less bins compared to that in Sturges' rule. If data does not follow the normal distribution, more bins may be needed for more skewed data [24] to preserve the shape, however, in our case more skewed data is actually easier for dense region detection (see Section 3.1).

A d-dimensional histogram can be constructed in a similar way, [2] each *bin* is a d-dimensional hypercube (e.g. for 2-dimensions (2-d), it is a square), *binwidth h* now refers to the width of each side of the hypercube: $\hat{f}^d(x) = \frac{1}{Nh^d}(\text{number of objects in the same bin as } x)$

[1]Before a histogram is built, the dataset must be pre-processed to remove any extreme points since such points can greatly distort the results, indeed one such point very far from the other normal data values can force all the normal values to be inside a single bin, jeopardizing the clustering results.

For simplicity we assume that the domain of each dimension is normalized so that the full space is a unit hypercube.

[2]Equisized or equi-width histograms are used here. In query optimization in RDBMS, histograms are used to approximate the distribution of data. There the equi-depth approach is in general the better choice. However, our purpose of using histograms is quite different from that in query optimization problems. Here we are interested to find the clusters and their locations. In the equi-depth approach, each partition in the histogram contains the same number of data objects. Let us consider a single dimension projection. Suppose we have two cluster projections on this dimension with a wide empty space in between. With equi-depth, we would likely end up with a partition that contains the middle space and together with quite a lot of data from both clusters. This partition will be considered sparse, and would be pruned away. However, along with the pruning we may also lose significant parts of the two clusters. This is the reasoning for the choice of the equi-width approach.

Each histogram is related to a subspace. For example, if we have three dimensions (attributes), A, B, C, and 2-d histograms are used, then the corresponding subspaces will be AB, AC, BC. We assume an arbitrary ordering of the histograms/subspaces, and we shall denote the subspaces by $S_1, S_2, ...$. For example, in the above case, S_1 equals AB, S_2 equals AC, S_3 equals BC. In a histogram, **dense regions** may be found.

Definition 5: A **dense region** $DR_m^{S_i}$ associated with subspace S_i is an intersection of intervals from the dimensions in S_i, in which the data density inside the intersection volume is larger than a certain threshold. This is intended to contain the projection from one or more projected clusters on the subspace. The **dense_region id** of the dense region is given by m.

Notice that we do not require the user to specify the above threshold value, it will be derived from the data set.

B. Detecting dense region

For discovering the dense regions, we use different thresholds along different subspaces, the threshold is driven by the data set distribution. In choosing an appropriate threshold value, we would like this value to be able to distinguish among the random noise/outliers and projections of the clusters. Clusters vary with their underlying distribution, spanning area, and number of data objects contained.

We explain our method to determine the thresholds with the help of a single cluster projection. Often a cluster is more dense in its center, and less dense near the boundary. A more irregular cluster can have irregular dense areas. An extreme case is when a cluster has uniform density. Consider two different cases in Figure 3(a). One projection of the cluster establishes a "bell" shape which is denser in the center, while another one form a "flat cluster" with constant density. Even though the two clusters contain the same number of data objects, the latter is more difficult to be discovered by a single threshold value. Threshold 1 can detect the "bell" cluster, although missing some data objects at the boundary, but it will miss the "flat" cluster. In general given projections of clusters with the same number of data points, projections with uniform density are more difficult to be detected, since there are no peak regions with a highest density in the projection. Therefore, we consider this worst case with a "flat" cluster in the following analysis.

Consider Figure 3(b), the total number of data objects is N, and the total number of histogram partitions (bins) in the subspace is H. From the cluster(s), there is a higher density projection in the subspace which contains a fraction F of the N data objects, where $0 \leq F \leq 1$. Assume that the projection spans a fraction f of H histogram intervals, where $0 \leq f \leq 1$. We establish a theorem which can tell us how to set the appropriate value of the threshold to detect dense projections, based on the mean μ, the standard deviation σ of the data distribution, and the users' expectation on how large a cluster's projection would span. Suppose the 1–d histogram for a dimension D is given by $\mathcal{X} = \{X_1, X_2, ..., X_p\}$. Let the mean value ($\frac{1}{p}\sum_{i=1}^{p} X_i$) be μ. The standard deviation σ is

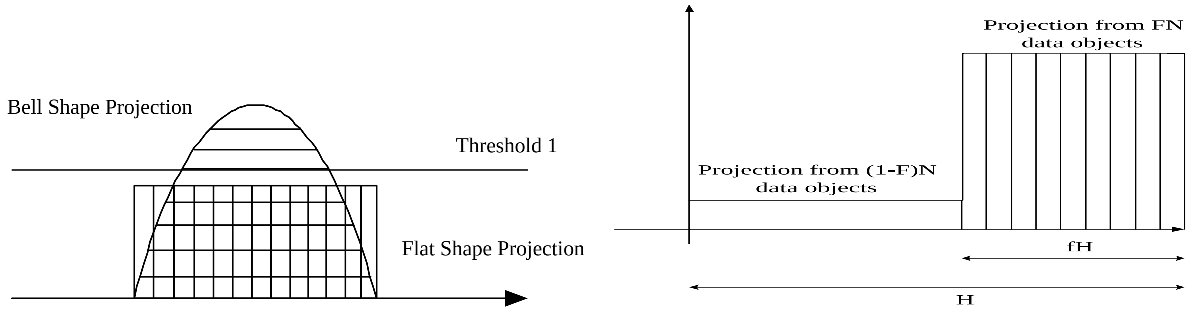
Bell Shape Projection
Threshold 1
Flat Shape Projection
(a)
Projection from FN data objects
Projection from (1-F)N data objects
fH
H
(b)

Fig. 3. Choice of the density threshold

given by $\sqrt{\frac{1}{p}\sum_{i=1}^{p}(X_i-\mu)^2}$. The **density** of a projection is defined as the number of data objects falling onto the projection divided by the number of intervals the projection spans.

Theorem 1: Let H be the number of d-dimensional histogram partitions (bins) in a subspace and suppose a projection of the data on this subspace consists of fH bins with uniformly high density, ($0 \le f \le 1$), and the remaining $(1-f)$ of the H bins have a uniformly low density. Let ρ be the higher uniform density, μ be the mean and σ be the standard deviation of the projection distribution. Then $\rho = \mu + \sqrt{\frac{1}{f}-1}\sigma$.

Proof: Without loss of generality, we move the region covered by the projection with the highest density to the right, leaving the low density bins to the left of the histogram, as in Figure 3(b). Let $(X_1,...,X_n)$ be the histogram values. The mean of the histogram values, μ, equals $\frac{N}{H}$. For the high density region, the uniform density ρ equals $\frac{FN}{fH}$, since there are fH partitions with high density. For the low density region, the density is $\frac{(1-F)N}{(1-f)H}$, since there are $(1-f)H$ partitions with low density. The square of the standard deviation σ can be expressed in terms of f, F, N, H as follows:

$$\begin{aligned}\sigma^2 &= \frac{\sum(X-\mu)^2}{H} \\ &= \frac{[(\frac{FN}{fH}-\frac{N}{H})^2 fH + (\frac{(1-F)N}{(1-f)H}-\frac{N}{H})^2(1-f)H]}{H} \\ &= \frac{N^2}{H^2}\frac{(F-f)^2}{f(1-f)}\end{aligned}$$

Now, we can derive a relationship of ρ with respect to μ, σ as follows:

$$\begin{aligned}(\rho-\mu)^2 &= (\frac{FN}{fH}-\frac{N}{H})^2 = \frac{N^2}{H^2}(\frac{F}{f}-1)^2 = \frac{N^2}{H^2}(\frac{F-f}{f})^2 \\ (\rho-\mu)^2 &= \sigma^2(\frac{1}{f}-1) \;\; ; \;\; \rho = \mu + \left(\sqrt{\frac{1}{f}-1}\right)\sigma\end{aligned}$$

■

Corollary 1: For any data distribution, if the projection of clusters on a d-dimensional subspace spans at most f of the bins in the corresponding histogram, then a threshold of ρ set greater than $\mu + \left(\sqrt{\frac{1}{f}-1}\right)\sigma$ can be used to detect the clustering, where μ is the mean and σ is the standard deviation of the projection distribution.

From Figure 3(b) if a threshold is set to be below ρ, then the high density projection will be detected. The above theorem gives the relation of ρ to the mean value and the standard deviation of the entire projection distribution. From the theorem, ρ can be expressed as $\mu + c\sigma$, and the value of c can be any value less than $\sqrt{\frac{1}{f}-1}$, where f is the spread of the high density region in the histogram projection. Note that the high density occurrence can be in multiple intervals, i.e. if there are a number of clusters and their projections in the histogram are not continuous, the theorem still applies. With this theorem, we can set the threshold to be lower than $\mu + c\sigma$ to detect the projections with the highest density in the subspace.[3]

For example, if the most spreading projections should span at most $\frac{1}{3}$ of partitions, we can set $c < \sqrt{\frac{1}{\frac{1}{3}}-1} = \sqrt{2}$. Note that Theorem 1 applies to histograms of any dimensions. For dimensions of 2 and above, a cluster may span much less than half the bins. This is because a cluster would typically span a fraction of the domain at each single relevant dimension and for histograms of higher dimensions, these fractions are multiplied which results in a much smaller fraction. [4]

In case when there are multiple projections on the subspace, we adopt an adaptive approach to iteratively lower the threshold value until no more dense regions are discovered.

Adaptive approach: To uncover multiple cluster projections with possibly different densities on a subspace, we adopt an adaptive approach to iteratively lower the threshold value until no more dense regions are discovered. Specifically, we set the initial threshold value to be $\mu + c\sigma$ according to Theorem 1. In the first iteration, projections with densities lower than the threshold would be treated as random noise or outliers. After detecting projections above the threshold, their densities and effects would be removed from the histogram. The new μ, σ, and the threshold would be calculated. We continue to detect and remove densities inside the dense regions, until no more dense regions are discovered, or the process has been repeated

[3]We emphasize more on being able to detect the existence of any cluster. The exact span of the cluster is not as big a problem. It is because in the projective clustering problem, detecting the proper subspaces for the clusters is a much harder problem than obtaining the clusters after a subspace is found. Many existing algorithms can handle the clustering problem once the subspace is identifi ed.

[4]In our implementation, we fi nd that $c = 2.5$ for 1-d histogram, and $c = 5$ for 2-d histogram can produce desirable results. If the user wants to determine f or c, he/she can visually examine some of the histograms to decide on a suitable value.

$max\_no\_cluster$ times. [5]

This idea is illustrated in Figure 4. In the figure, the dotted lines indicate the threshold values used in the iterations. We use the label DRi to label the i-th Dense Region sorted by the starting positions of the dense regions. In the example, we can also see another detail in the algorithm where adjacent dense regions are *merged* to form bigger dense regions. For example in Iteration 1, the 5th interval is considered dense; in Iteration 2, intervals 3, 4, and 6, 7 are considered dense. So in Iteration 2, a bigger dense region of $DR1$ is formed which ranges from interval 3 to interval 7.

Theorem 1 helps to explain why our method works well in many cases. A natural cluster typically does not span the entire domain when projected to some low dimensional subspace. Theorem 1 shows that when projected to one dimension, as long as the cluster does not span more than half the domain, the density threshold of $\mu + \sigma$ is able to detect it. If the cluster spans less of the domain, a higher threshold can be set accordingly.

C. Compressing the data objects and deriving their signatures

In the context of projective clustering, we are interested to discover dense regions to which the data objects would belong when projected onto each subspace corresponding to a histogram. Any unnecessary details can be pruned away. In order to do so, we compress the data objects from a D-dimensional vector to a **signature** with ${}_DC_d$ entries, where d is the dimensionality of the histograms we construct in the previous phase. With these signatures, we could easily group "similar" data objects from the same projected cluster. (Note that the signature concept is similar to the IdString in [2] as described in Section 2.)

Definition 6: Given subspaces $S_1, S_2, ...S_k$, where $k = {}_DC_d$, which are the subspaces for the histograms. A **signature** Q_i for data object O_i is an ordered list of ${}_DC_d$ entries, where the j^{th} entry represents the dense region, if any, where the data object is located in subspace S_j. Specifically, $Q = [Q_1, Q_2, \ldots, Q_L]$

$$\text{where } Q_j = \begin{cases} 0 & \text{if the object does not fall into} \\ & \text{any dense region in subspace } S_j, \\ r & \text{if the object is located in dense region} \\ & DR_r^{S_j} \text{ in subspace } S_j. \end{cases}$$

Each non-zero entry corresponds to a **bounded subspace**, and each zero entry corresponds to an **unbounded subspace**. From the signature, we can estimate the associated subspace of the projected cluster, in which the corresponding data object is likely to belong to. Here we call the estimated associated subspace a **derived subspace** S, and the actual subspace of cluster that the data object should belong to as the **associated subspace**.

[5]Note that in the adaptive approach, the same c is used for the density threshold in all iteration. We explain here the reason. In each iteration, we remove some parts that are are of higher density, and the remaining parts of the histogram contains more of the sparse portions (that means parts that are not the projection of clusters). Hence the projection of clusters on the remaining parts will be more skewed, or the portion should be less than before. Since c is for an upper bound on this portion, it can be reused.

Example 1: Suppose there are four dimensions *A, B, C, D*, and we have constructed ${}_4C_2 = 6$ 2-d histograms. The signatures should contain 6 entries, where the first, second ... entries corresponds to subspaces *AB, AC, AD, BC, BD, CD*, respectively. For example, suppose that for object O_1, the signature is [2 0 0 1 0 0]. Its bounded subspace is *AB* and *BC*. We may estimate the associated subspace of cluster the data object belongs to (the derived subspace) as *ABC*. However, it seems that if the associated subspace is *ABC*, all *AB, BC, AC* should be the bounded subspace. There are two possible reasons. (1) *ABC* is not the associated subspace. (2) *AC* should also be a bounded subspace, but O_1 is located at the boundary of the projection on *AC*, in which the location is not considered as dense.

Suppose we have an object O_2 whose signature is [2 3 0 2 0 0]. Its bounded subspaces are *AB, AC* and *BC*. Its derived subspace is *ABC*. This time, since all 2-d subsets of *ABC* are bounded subspace, we are quite sure that the derived subspace should match the associated subspace. Comparing these 2 cases, we say that the derived subspace *ABC* is of **high confidence level** for O_2, and it is of *lower confidence level* for O_1. ■

We estimate the derived subspace of a signature, by taking the union of all corresponding bounded subspace.

Definition 7: We define the **confidence level** of a derived subspace with l_j dimensionality, estimated by d-dimensional histograms for data object O_i as:

$$\frac{\text{number of bounded subspaces in } S \text{ in signature of } O_i}{\binom{l_j}{d}} \quad (1)$$

This corresponds to the ratio between the number of bounded subspace in S in the signature of the data object, and the number of all possible subspaces of S of d dimensions. When this ratio approaches 1, the derived subspace should be an accurate estimation of the associated subspace.

Two signatures having the same derived subspace can be in two different projected clusters, if they are in different projections in some bounded subspace. Consider Example 1, the dense_region ids of bounded subspaces *AB, BC* of O_1 are 2, 1, respectively; the dense_region ids of bounded subspaces *AB, AC, BC* of O_2 are 2, 3, 2, respectively. Both data objects $O_1 and O_2$ have the same derived subspace *ABC*. However they should be in two different projected cluster, since they share a common bounded subspace *BC* but with different dense_region ids – O_1 is in DR_1^{BC} while O_2 is in DR_2^{BC}. We call the common bounded subspace of two data objects **conflicting** if they have different dense_region ids in this bounded subspace.

To compress the original data set, we compute the signature and the derived subspace for each data object. We create a data structure called the **signature list**. Every entry in the signature list records a derived subspace S, and the dense_region id in each bounded subspaces in S. This should correspond to a potential cluster region C. Signatures of objects with the same derived subspace and no conflicting dense_regions in bounded subspaces would be combined and inserted into the same entry of the list.

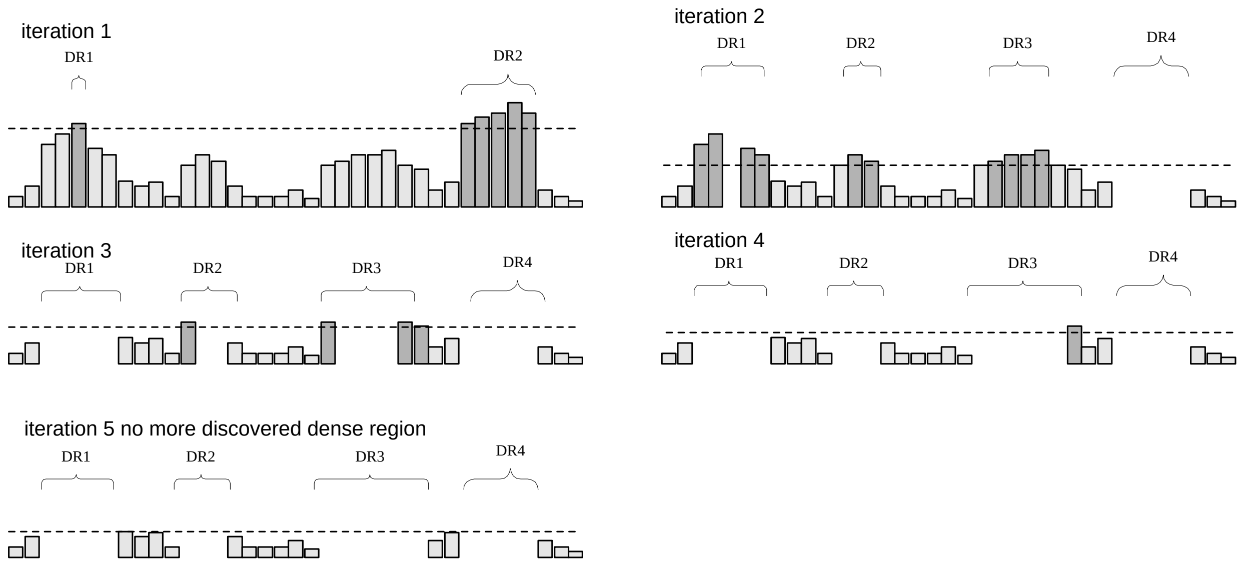

Fig. 4. Adaptive approach to iteratively lower the threshold value.

In addition a weighting for each entry of the signature list is computed as follows:

Definition 8: The **weighting** of an entry of the signature list is calculated as the summation of the confidence level of the data objects contributing to this entry.

We can explain the weighting by the membership of each data object related to the cluster region C. If the confidence level is high, the object has high membership and should be counted with more weight. If confidence level is low, it is possible that the object does not really belong to the cluster region, and should be counted less.

D. Merging Similar Signature Entries

Each entry in the signature list corresponds to a group of data objects with the same derived subspace and sharing no conflicting bounded subspaces. Most likely, this group of data objects are from the same projected cluster. However, there are often cases where data objects from the same projected cluster would have different derived subspaces, and thus contribute to different entries in the signature list. Consider Example 2.

Example 2: Suppose there are four dimensions *A, B, C, D*. The signatures contain 6 entries, where the first, second ... entries corresponds to subspace *AB, AC, AD, BC, BD, CD* respectively. Consider data object sets O_1, O_2, O_3, O_4, whose signatures are [2 0 0 3 0 0], [2 0 0 1 0 0], [2 0 1 0 3 0] and [2 0 1 0 3 2] respectively, and their derived subspace are *ABC, ABC, ABD, ABCD*. Although O_1 and O_2 have the same derived subspace, they are inserted into different entries in the signature list because they have the conflicting subspace BC. Data sets O_1 and O_2 are totally dissimilar.

Data sets O_3 and O_4 have different derived subspaces and are in different entries, but they share many common bounded subspaces. They can be in the same projected cluster if O_3 falls in the boundary of the projection on *CD*, or O_4 occasionally falls in the projection on *CD*. We consider O_3 and O_4 to be similar.

Data sets O_2 and O_4 have different derived subspaces, and they share only one common bounded subspace in the total of 5 bounded subspaces. Sets O_2 and O_4 are not similar. ■

We want to merge "similar" signature entries, as they may correspond to groups of data objects located in the same projected cluster. We adopt the following similarity measurement.

Definition 9: Consider the signatures s_i and s_j of two data sets O_i and O_j, respectively. Suppose the number of unique bounded subspaces from s_i and s_j is A, and there are B common bounded subspaces in s_i and s_j that share the same dense region ids. The **similarity** between the two signature entries is given by

$$S(O_i, O_j) = \frac{\text{number of common bounded subspaces}}{\text{total number of unique bounded subspaces}} = \frac{B}{A} \quad (2)$$

In Example 2 above, $S(O_1, O_2)$ equals $\frac{1}{2}$ and $S(O_3, O_4)$ equals $\frac{3}{4}$. Signature entries sharing large proportion of common bounded subspaces would have a high similarity value and can be considered as similar. When two signatures are merged, the signature with a bigger set of data points assigned serves as the merged signature, breaking ties arbitrarily. Let m be $max\_no\_cluster$. To speed up the merging process, we keep only the $k \times m$ top signature entries ranked by the weightings, where k is a parameter depending on the available resources. Signatures with low weightings would be pruned away. In current implementation, we set k to be 50. [6] If the user have some knowledge about a rough estimate of the amount of noise/outliers, the pruning of signatures can be based on this amount, since the pruned signatures should correspond to the noise/outliers in the data.

Similarity of every pair of the remaining signature entries would be computed according to Equation 2. We continually combine the pair of signature entries with the highest similarity

[6]We note that $k = 50$ gives good results in terms of clustering for different data sets in our experiments and the effi ciency is high. Hence we suggest to use this as a default value.

until reaching a termination criteria. One termination criterion is when the number of signatures reaches *max_no_cluster*. However, if the signatures are quite different before we reach *max_no_cluster*, then the merging can be stopped when we see a sudden drop [7] of the highest similarity among the signature, or the highest similarity is already less than a lower bound threshold (in current implementation, we use a threshold of 0.3). [8] We do a final sorting in descending order of weighting. The remaining signature entries correspond to the clusters and subspaces detected by the algorithm. Note that we may end up with more than *max_no_cluster* clusters at this point with the second termination criterion.

Next, at most $max\_no\_cluster$ signature entries with the highest weighting would be kept after merging: if there are more than $max\_no\_cluster$ entries then the top $max\_no\_cluster$ signature entries with the highest weightings are kept; otherwise all the signature entries are kept.[9] Each signature entry corresponds to a projected cluster, where its derived subspace represents the associated subspace, and the recorded dense_region id locates the cluster on the subspace.

E. Associating membership degree

The above discussion has been based on hard clustering. For fuzzy clustering, each data object O_m is associated a membership degree for each discovered projected clusters C_n:

$$member(O_m, C_n) = \frac{\text{number of matched bounded subspaces}}{\text{number of bounded subspaces in the signature list of } C_n} \quad (3)$$

Equation 3 measures the degree of membership in terms of similarity between the signature of the data object and the cluster. If this measurement approaches 1, that means the data object is located in all dense regions projected by the cluster. Thus, it has the strongest membership degree. On the other hand, if this approaches 0, that means the data object is not located in any dense regions projected by the cluster.

For hard clustering, we first compute the membership functions, then a data object would be assigned to the output cluster with the largest degree of membership. If users expect outliers to be uncovered, a certain threshold can be set so that a data is considered as outlier if its membership degrees are all below this threshold (e.g. in current implementation we use 0.1). Another way is to specify an expected percentage of outliers, such an amount of data with the lowest degrees of membership will be considered outliers. Still another way is by a combination of outlier percentage and the membership values.

[7]The sudden drop of highest similarity is only a suggestion for a stopping criterion. A possible definition of sudden drop is by means of the Minimum Description Length, as described in [2] for pruning uninteresting subspaces. However, we have not really used this suggestion in our implementation.

[8]We have tried this threshold with a number of data sets and it can give good results.

[9]Let w be the weighting of the $max\_no\_cluster$-th cluster in the ranking by weightings. If there exist more than one cluster with weighting w, then we can remove all such clusters, in order not to exceed the bound of $max\_no\_cluster$. If the number of resulting uncovered clusters differs greatly from the estimated $max\_no\_cluster$, the system could report this to the user, since the user may want to know that he/she has greatly underestimated or overestimated the number of clusters.

F. The choice of Dimensionality d of the Histogram

User may not be able to choose the value of d which gives satisfactory cluster result and acceptable computational time. Here we propose a framework in which a user does not need to select the d value. First we perform a very fast clustering using 1-d histograms. If the 1-d histograms cannot detect and distinguish clusters well, data objects would be associated with a low degree of membership, in which many of them would be outputted as outliers (or noise). We record the detected clusters and partitioned data objects. The unpartitioned data objects (which are outputted as outliers in this run) would be input to the next run, in which we do clustering using 2-d histograms. Precaution can be made to uncover any cluster from 2-d which can be merged with some cluster from 1-d. We may continue this approach with higher dimensionality histograms, until the number of remaining outliers (or noise) is less than a certain fraction of data objects (e.g. 5% of the whole data set). Note that the complexity of the computation increases with the choice of d, and which would also be a factor to consider. From our experiments, the 2-d histograms are highly effective for the tested datasets, we believe that they are also sufficient for most applications.

IV. The Implementation

Our implementation of EPCH follows our discussion in the previous section. EPCH starts by constructing d-dimensional histograms to model the data distribution where d is adjustable according to users' expectation on the clustering quality and the running time. When we set d to 1, it is similar to EPC [19] which uses 1-d histogram. We choose d to be 2 in the following discussion for implementation to illustrate the idea. We call this implementation EPC2. The same approach can be applied for other values of d. The overall algorithm consists of five phases.

1) **Histogram Building phase** builds ${}_DC_d$ histograms, each corresponding to one d-dimensional subspace. For EPC1, we set the number of bins in each subspace to be 200. In EPC2, we set the number of bins to be 400.
2) **Dense Region Detection phase** works iteratively to identify all dense regions.
3) **Signature List Construction phase** examines each data object, and derives its signature according to the *id* of the dense regions where the object is located in the subspace for each histogram. From the signature, we find the derived subspace by the union of all bounded subspaces, and its confidence level according to Equation 1.
4) **Merging similar Signatures phase** sorts the signature entries from the previous step according to the descending order of their weightings, where a signature entry with higher weighting corresponds to a cluster with more data objects and clearer associated subspace.
5) **Membership Degree Assigning phase** associates each data object and discovered cluster with a degree of membership according to Equation 3.

A. Time and Space Complexity

Let N be the number of data points, $m = max\_no\_cluster$, D be the number of dimensions, d be dimensionality of histograms, H be the number of bins in each histogram, l be the average dimensionality of the associated subspaces of clusters, and km be the number of signatures kept in the merging step. The total time complexity is $O(ND^d + mHD^d + N(l^d + D^d) + Nlog(km) + (km)^2 l^d + Nml^d)$. If we consider $log(km)$ small, this can be simplified to $O(D^d(N + mH^d) + l^d((km)^2 + Nm)$. The space complexity is $O(HD^d + NDl^d + Nm)$. In particular, for EPC2, consider H small, and d is fixed to be 2, the time complexity is thus $O(N(D^2 + l^2 m) + (km)^2))$. For 1-d histograms, d equals 1, again assuming H is small, the time complexity is $O(N(D + lm) + l(km)^2)$.

V. Experimental Results

We present several experimental results and their analysis in this section. We have implemented PROCLUS, ORCLUS, EPC1 (EPCH with 1-d histograms) and EPC2 (EPCH with 2-d histograms), and would like to compare their performance differences in terms of clustering quality and running time. Experiments are performed on synthetic data and a set of real data.

A. Clustering Quality Measurement

Different quality measurements have been used in previous work on projective clustering. Confusion matrix is used in [4], [5]. *Dominant ratio* is suggested by [3]. In [4] the dimensions of the subspaces found and those of the clusters are compared. We adopt all these measurements, and also added our own suggestion of the coverage ratio to complement some insufficiency.

- **Confusion Matrix** is often used to evaluate the clustering result of synthetically generated data. Specifically, it is a $p \times p$ matrix, where p is the number of natural clusters. Entry (i, j) records the number of data objects belonging to the natural cluster i, that is assigned to the output cluster j. Obviously, if we can observe a clear one–to–one mapping between each output cluster to a natural cluster, the clustering quality is good. The left hand side of Table I shows an example of good clustering result, and the right hand side represents a bad clustering result.
- **Dominant Ratio**: [3] suggested a measurement *dominant ratio* to evaluate the quality of the clustering. This is the average fraction of data objects in each output cluster which are populated by the most dominant natural cluster. For each output cluster, we identify the natural cluster which contains the largest number of data objects. Then, we calculate the percentage of data objects in the output cluster which was populated by this natural cluster. Averaging such values over all output clusters yields the dominant ratio. A good clustering should have the dominant ratio close to one.
  Here, we argue that this single measurement may not be able to justify a good clustering. Consider an extreme case where there are many output clusters so that each natural cluster becomes a number of output clusters. The inappropriate splitting of natural clusters could still produce a dominant ratio of 1.
- **Coverage ratio**: Therefore, in addition to dominant ratio, we propose another measurement called the *coverage ratio*, which in a way complements the dominant ratio. This is the average fraction of data objects in each input cluster which are covered by the most dominant output cluster. For each natural cluster, we identity the output cluster which contains the largest number of data objects. The percentage of data objects in the natural cluster which are populated by their dominant output cluster is calculated, which would be averaged over all natural clusters to yield the coverage ratio. (Note that with this definition we give similar importance to each cluster, independent of its size.) Similarly, there would be cases where an output cluster combines several natural clusters and still give rise to a coverage ratio close to 1.

We can evaluate the effectiveness of a clustering algorithm by the resulting dominant and coverage ratio. A good clustering result should yield both dominant ratio and coverage ratio close to 1. In this case, all output clusters can cleanly map to natural clusters, and no output cluster combines several natural clusters; at the same time, no natural clusters are split by several output clusters. For a clustering algorithm which often splits natural clusters, the dominant ratio would be much higher than the coverage ratio. On the other hand, if the coverage ratio is significantly larger than dominant ratio, the clustering algorithm often combines natural clusters.

We have measured the performance in our experiments with these three measurements. We shall report our results in one or two of the measurements in each set of experiment in the following.

B. Synthetic Data Generation

We generate data sets with different properties.

- **PR-Set**: follows data generation in [4] (PROCLUS), where both the vectors forming the subspace and the spreading directions of the clusters are parallel to the original axes (as shown in Figure 2a). Associated subspaces of clusters are with varying dimensionality. Data objects follow the normal distribution in bounded dimensions with small variance. Clusters in this data set should be easier to be detected, as its projections to any single bounded dimension is very dense.
- **AP-Set**: models clusters associated with arbitrary subspaces (as shown in Figure 2b). It follows the data generation described in [5](ORCLUS) with some modification so that the bounded subspaces are not determined by all of the original dimensions (see discussion in Section 1). Specifically, dimensionalities of all associated subspaces are set to a fixed value l. For each cluster, we choose randomly l dimensions as its bounded dimensions, and then generate l orthogonal vectors randomly in the subspace formed by the bounded dimensions (by finding eigenvectors in a randomly generated symmetric matrix). The l eigenvectors form the orientations of the clusters. Anchor point of each cluster would be randomly chosen. Data objects distribute along the vectors defining the orientation in its associated subspace, following normal

distribution with their mean at the anchor point. In other unbounded dimensions, they distribute randomly. As in [5], we set l to be 6 in the base case.

In real life application, data often contains outliers (data objects do not belong to any cluster, or random noise), and dimensionalities of different associated subspaces need not be the same. Therefore, we generated two variations on the AP-Set.

1) **APN-Set** is the same as AP-Set which includes a certain percentage of outliers. (we use 5 % of noise in the base case.)
2) **APD-Set** contains clusters with associated subspaces with varying dimensionality. In particular, for data sets with 5 clusters, we set the dimensionalities of associated subspaces of clusters to be 4, 5, 6, 7, 8.

We use these two data sets to evaluate performances of different algorithms on data with these two properties.

C. Experimental setup

All the experiments have been performed on a 12 UltraSPARC-II 400 MHz machine with 8GB RAM, running Solaris 7. The algorithms are implemented using C language and gcc compiler v2.7 without code optimization. For PROCLUS implementation, we set the number of seed points to be 2% of data objects, and number of iterations allowed for no quality improvement to be 20. For ORCLUS implementation, we use the parameter values suggested in [5], except that we set the reduction factor α of the number of clusters in each iteration be 0.8 instead of 0.5 to obtain a more accurate clustering result. For data set with outliers, we follow suggestions in [5] to add outlier handling implementation, and name it as ORCLUS_outlier. For implementation of EPC1, we set the threshold to detect the dense regions to be $\mu+2.5\sigma$. For EPC2, we set the threshold to $\mu + 5\sigma$. [10] We report here some special properties of each clustering algorithm and general trends. It is found that the behaviour of hard clustering and fuzzy clustering are similar in our experiments. We focus our attention on hard clustering in the discussion.

1) Comparison between EPCH and PROCULS: Since both PROCLUS and EPC1 are targeted to discover clusters in PR-Set, and both can report the bounded dimensions of the associated subspace, we evaluate their clustering qualities, the accuracy of identification of subspaces, and the running time in this data set. We vary N from 3000 to 200000, and set $D = 20$. In EPC1 we set the number of bins in each histogram to 200, which is much greater than $1 + log_2 N$. (If $D = 1 + log_2 N$ is adopted, then $D \leq 20$ would be sufficient for all test cases.) In order to evaluate how many clusters are correctly discovered, and the number of correctly uncovered bounded dimensions in the corresponding associated subspaces, we set the criteria of clear ”one-to-one” mapping in the resulted confusion matrix as follows:

Output cluster X has a clear ”one-to-one” mapping to natural cluster Y, if more than 60% of data objects from Y are reported to belong to X, and X contains no more than 20% of data objects from other natural clusters. Based on this criteria, a natural cluster is said to be correctly discovered if we can find a clear ”one-to-one” mapping between this cluster and an output cluster from the confusion matrix. Due to the limited space, we show in detail only one of the results obtained from a data set with N = 6000 in Table I. Other data sets show similar trends. The shaded field corresponds to the correctly discovered clusters. In this data set, EPC1 can discover 4 clusters. One of the good property of EPC1 is that it always returns clusters containing the largest number of data objects first. It is not surprising that EPC1 cannot discover the smallest cluster, as it contains less than 5% of data objects, which may be considered as random noise. For PROCLUS, it can discover only 2 clusters.

	EPC1		PROCLUS	
Data Set	dominant ratio	coverage ratio	dominant ratio	coverage ratio
3000	1.000	0.908	0.812	0.717
6000	0.998	0.864	0.764	0.763
10000	1.000	0.505	0.755	0.668
20000	1.000	0.912	0.732	0.316
30000	1.000	0.902	0.838	0.469
40000	1.000	0.745	0.764	0.715
65000	1.000	0.906	0.629	0.739
80000	1.000	0.783	0.863	0.838
100000	0.993	0.736	0.930	0.163
200000	1.000	0.912	0.732	0.316

TABLE III
DOMINANT AND COVERAGE RATIOS OF RESULTS OBTAINED EPC1 AND PROCLUS WITH DATA SETS VARYING N

We calculate the percentage of correctly partitioned data objects, and also compare the reported dimensions in the associated subspaces of the output clusters, with the associated subspace of the corresponding matching natural clusters. Table II compare the results obtained from EPC1 and PROCLUS in PR-Set.

Table III shows the dominant and coverage ratios of these two algorithms. We can see that EPC1 produces more accurate result than PROCLUS in nearly all cases. Figure 5 compares their running time with varying N. Both methods scale about linearly with N. EPC1 is much faster than PROCLUS. For example, for the data set with $N = 10000$, EPC1 takes 20 seconds and PROCLUS takes about 10 minutes to compute. Since EPC1 is more accurate and efficient than PROCLUS, in the following discussions, we mainly compare EPC2, ORCLUS, and EPC.

D. Comparison between EPCH and ORCLUS

We compare the differences among clustering qualities of the AP-Set, APD-Set, APN-Set and PR-Set. We observe that varying the number of data objects would not cause any significance change on the clustering quality. We ran experiments for different datasets with N ranged from 3000 to 60000, and obtain similar general trend for dominant and coverage ratios. Table IV shows the average of the dominant and coverage ratios over data sets with varying N, for AP-Set, APD-Set and PR-Set. Note that for AP-Set ORCLUS was

[10]We found that these threshold settings are quite robust to changes in different data sets, and hence we suggest that they are used as default values.

natural output	Outlier	0	1	2	3	4
Outlier	302	120	156	83	122	35
0	0	0	0	0	3168	0
1	0	0	0	6	0	819
2	0	0	0	721	0	0
3	0	0	468	0	0	0
4	0	0	0	0	0	0

(a)

natural output	Outlier	0	1	2	3	4
Outlier	61	4	0	0	69	12
0	25	0	0	810	0	207
1	55	34	624	0	0	102
2	33	3	0	0	0	267
3	103	67	0	0	122	215
4	25	12	0	0	3099	51

(b)

TABLE I
CONFUSION MATRIX OBTAINED FOR PR-SET WITH 6000 DATA OBJECTS FROM (A)EPC1, (B)PROCLUS.

	EPC1			PROCLUS		
Data Set	Correctly discovered clusters	Percentage of correctly partitioned data objects	Correct dimension/ Total dimension in the correct clusters	Correctly discovered clusters	Percentage of correctly partitioned data objects	Correct dimension/ Total dimension in the correct clusters
3000	4	86	19/19	2	72	6/10
6000	4	94	18/18	2	65	7/10
10000	5	84	25/25	3	88	7/14
20000	4	92	20/20	0	0	0/0
30000	5	93	25/25	2	39	10/12
50000	3	94	15/15	2	81	6/10
65000	4	94	19/19	3	91	11/15
80000	4	94	17/19	4	88	16/20
100000	3	78	13/13	0	0	0/0
200000	5	90	25/25	2	75	8/12

TABLE II
SUMMARY OF THE COMPARISON OF RESULTS OBTAINED FROM EPC1 AND PROCLUS FOR PR-SET

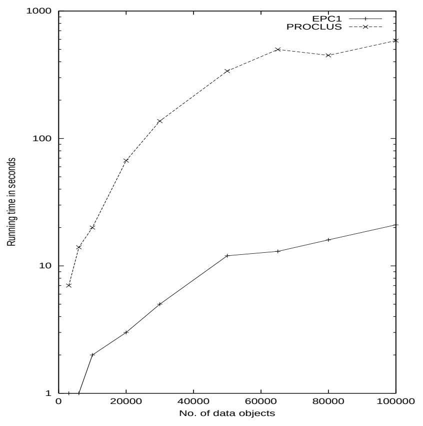

Fig. 5. Running time against N for EPC1 and PROCLUS

given the additional but unrealistic advantage that the actual dimensionalities of the clusters are assumed to be the same for all clusters and assumed to be known and provided to the algorithm, this cannot be true in general, since a user may not be able to know the parameter in advance and also the dimensionalities of different clusters are likely to be different.

1-d versus 2-d Histograms: Both EPC1 and EPC2 perform very well in PR-Set. For other data sets where the spreading directions of clusters may not be parallel to the original axes, EPC2 clearly outperforms EPC1. 1-d histogram can help to detect and distinguish projections from clusters with axes parallel spreading. For clusters with arbitrary spreading direction, using 2-d histograms or higher dimensionality can significantly improve the clustering quality. This is because with projection on a single dimension, overlappings of cluster projections may occur, which may result in a spread of the clustering projection that is close to uniform distribution, and in turn conceal the clusters. With 2-dimensional or higher dimensional histograms, the overlappings of cluster will have much lower chance to create a close to uniform distribution.

Associated subspaces with varying dimensionality: Comparing AP-Set and APD-Set, ORCLUS performs better in the AP-Set, because it assumes that the associated subspaces are with the same dimensionality, while APD-Set contains subspaces with varying dimensionality. In such case, for associated subspaces with lower dimensionality, ORCLUS would include additional unbounded subspaces and retain some of the noise from the data. For the same reason, ORCLUS does not perform well in PR-Set, which contains associated subspaces

	EPC2		EPC		ORCLUS	
	dominant ratio	coverage ratio	dominant ratio	coverage ratio	dominant ratio	coverage ratio
average over AP-Set	0.829	0.736	0.321	0.776	0.836	0.855
average over APD-Set	0.841	0.722	0.320	0.945	0.766	0.652
average over PR-Set	0.914	0.936	1	0.828	0.800	0.585

TABLE IV

AVERAGE DOMINANT AND COVERAGE RATIOS FOR AP-SET, APD-SET, AND PR-SET

D	EPC2		ORCLUS	
	dominant ratio	coverage ratio	dominant ratio	coverage ratio
10	0.882	0.819	0.750	0.768
20	0.846	0.811	0.830	0.739
30	0.843	0.631	0.662	0.647
40	0.941	0.920	0.537	0.593

TABLE V

DOMINANT AND COVERAGE RATIOS FOR APN-SET WITH $N = 20000$ AND $l = 6$

l	EPC2		ORCLUS	
	dominant ratio	coverage ratio	dominant ratio	coverage ratio
2	0.821	0.763	0.525	0.468
4	0.842	0.630	0.627	0.622
6	0.846	0.811	0.830	0.739
8	0.878	0.817	0.942	0.985
10	0.959	0.967	0.692	0.855

TABLE VI

DOMINANT AND COVERAGE RATIOS FOR APN-SET WITH $N = 20000$ AND $D = 20$

with varying dimensionalities. In contrast, EPC2 does not make this assumption. It gives satisfactory result in the APD-Set. The clustering quality of APD-Set is even slightly better than AP-Set, because the more variant the dimensionalities of the associated subspaces are, the easier it is for the signature merging phase to distinguish different associated subspaces by different signatures. The behavior of APN-Set is reported in the next subsection.

Dimensionality of the original space and the associated subspace

We observe an interesting phenomenon that ORCLUS has advantages when the difference between the dimensionality of the original space and the dimensionality of the associated subspace is small, i.e. the dimensionality of associated subspaces approach dimensionality of the original space. Performance of ORCLUS degrades when the difference becomes large. On the other hand, the accuracy of EPC2 does not vary much with different dimensionalities of the original space, and different dimensionalities of the associated subspaces. This happens in both AP-Set and APN-Set. Table V shows the dominant and coverage ratio with different values of D for APN-Set with $N = 20000$ and $l = 6$. Table VI shows the dominant and coverage ratio with different values of l for APN-Set with $N = 20000$ and $D = 20$.

From Table V, we observe that as D increases, the accuracy of ORCLUS decreases. Also, as shown in Table VI, ORCLUS can produce good clustering quality only when $l >= 6$. This is because the number of iterations in ORCLUS is determined by the ratio between number of seeds and number of clusters, and a reducing factor. Each iteration will reduce both the number of clusters and the dimensionality of the clusters. If the difference between dimensionality of the associated subspace and the original space is large, each iteration will need to "peel off" a great factor of the dimensionality of the current clusters. This decreases the accuracy in uncovering the subspaces. EPC2 is not affected by this factor. Therefore, EPC2 shows more advantages when the dimensionality of the original space is high, and at the same time the dimensionality of the associated subspaces is much lower. We believe that many real data sets have such property, as the database can contain up to hundreds of attributes, but patterns typically exist within small subsets of attributes.

E. Scalability

We plot the user CPU time on the three algorithms in APN-Set in Figure 6. Figure 6(a) plots time against varying N with D = 20 and l = 6. Figure 6(b) plots time against varying D with N = 50000 and L = 6. Figure 6(c) plots time against varying L with N = 50000 and D = 20.

We observe that EPC1 is extremely fast compared to ORCLUS and EPC2 in all cases. (e.g. for the largest data set, EPC1 requires less than 2 user CPU seconds, while EPC2 requires about 300s and ORCLUS takes about 1500s.) We examine the actual running time of different phases in EPC2, and observe that it is dominated by the merging signatures phase. The theoretical running time complexity of this phase is $O(Nlog(N) +^3 l^d)$, however, the average running time greatly depends on the similarity among data objects. The more is the dissimilarity among the objects, the less is the time required for the merging process to reach the termination criteria. Therefore, we cannot observe a clear trend of the running time with varying N. In any case, EPC2 is more efficient than ORCLUS when N is up to 65000.

With varying D, ORCLUS scales superlinearly, which agrees with the running time analysis that ORCLUS is $O(D^3)$ in [5]. It is not very feasible to apply ORCLUS to database with high dimensionality. EPC1 is about linear with D. EPC2 scales quadratically with D, which matches the analysis that the time complexity is $O(D^d)$ in Section IV-A. With varying l, both EPC1 and ORCLUS scale linearly. EPC2 scales superlinearly with l, because it takes longer time to merge two signature entries if their derived subspace is with higher dimensionality.

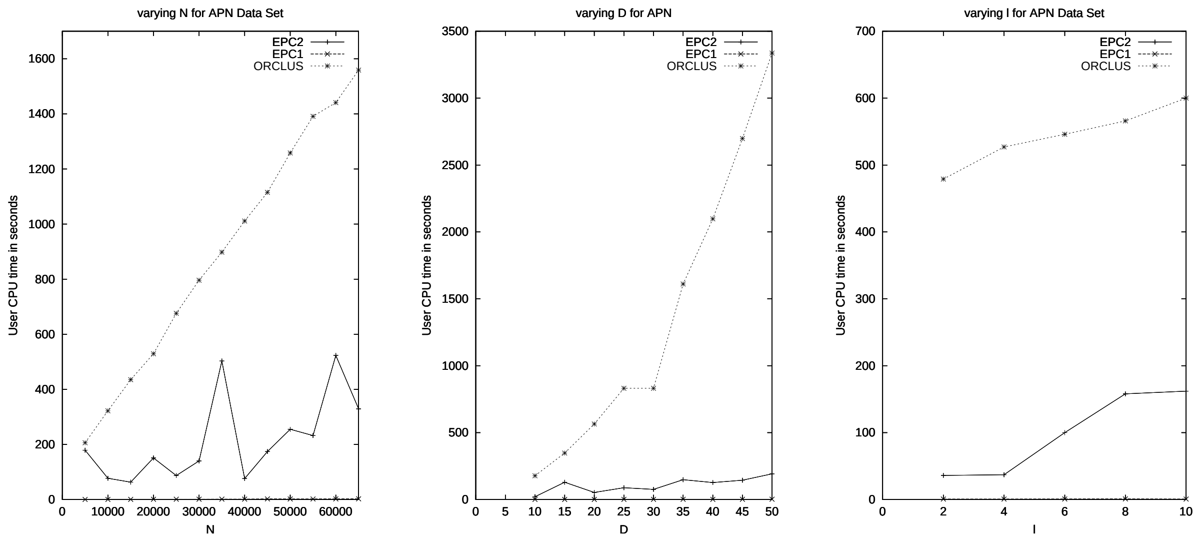

Fig. 6. User CPU Time against a)varying number of data objects, b)varying original dimensionality, c)varying dimensionality of associated subspaces

F. Real Data

We conducted the experiment on a real data set called Image Segmentation data [1]. The instances in the data set were drawn randomly from a database of 6 classes of outdoor images. There are totally 180 records and 19 numerical attribute in the data set. The data set contains 6 classes, namely sky, foliage, cement, window, path and grass. We have pre-processed the data set. The values for each attribute are normalized. Extreme values in attributes are replaced with non-extreme values. One class of the data set has been removed because it has no resemblance to a cluster. We removed two attributes called short-line-density-5 and short-line-density-2 because these two attributes give the same value for most records.

We compare EPC2 with ORCLUS. The setting of EPC2 is the same as before. For ORCLUS, the targeted dimensionality is 20. The resulting confusion matrices are shown in Table VII. The dominant ratio and coverage ratio are shown in Table VIII. In measuring the results, we assume that the 6 classes above corresponds to 6 clusters. This assumption may not be valid. The data set may not possess clean clusters according to the nature of the classes since we can expect characteristics for two or more classes to overlap, for example, window may contain characteristics from all other classes, since a window can reflect their images, cement and path may share similar characteristics, similarly for foliage and grass. We found that EPC1 and PROCLUS did not produce good results. From Tables VII and VIII, we can see that EPC2 performs better than ORCLUS in all three metrics. From Table 7(a), the result is very reasonable or expected from the nature of the data set, we can easily deduce a cluster for sky, one for foliage, one for grass, and one for cement together with path. The result from PROCLUS is not as expected.

l	EPC2		ORCLUS	
	dominant ratio	coverage ratio	dominant ratio	coverage ratio
20	0.510	0.620	0.470	0.507

TABLE VIII
DOMINANT AND COVERAGE RATIOS FOR REAL DATA SET

VI. CONCLUSION

For high dimensional data, projective clustering is often more relevant than traditional clustering in the full space. Several algorithms have been proposed to solve the projective clustering problem, among those ORCLUS solves a more general problem. However, the complexity of ORCLUS is high, and it is not very feasible if the original dimensionality is high. ORCLUS makes the assumptions that the dimensionality of subspaces for the clusters are the same and also provided by user, both assumptions are hard to realize in applications.

We propose a new method for projective clustering by means of histograms. We call this method EPCH (Efficient Projective Clustering by Histograms). The advantages of EPCH include efficiency, effectiveness, and no assumption made about knowledge of the number of clusters or fixed dimensionalities for the subspaces.

Acknowledgements We thank Lai Mei Chiu for her help in the data generations and experiments. We thank the anonymous reviewers for their valuable comments and suggestions. This research was supported by the RGC Earmarked Research Grant CUHK 4179/01E of the HKSAR and the Innovation and Technology Fund ITS/69/03 of the HKSAR.

REFERENCES

[1] Uci machine learning repository. In *http://www.ics.uci.edu/ mlearn/MLRepository.html.*
[2] C. Aggarwal. A human-computer cooperative system for effective high dimensional clustering. In *Proc. of ACM SIGKDD Conference*, 2001.
[3] C. C. Aggarwal and P. S. Yu. Redefi ning clustering for high-dimensional applications. In *IEEE Transactinos on knowledge and Data Engineering, Vol. 14, No.2,* 2002.
[4] Charu C. Aggarwal, Cecilia Procopiuc, Joel L. Wolf, Philip S. Yu, and Jong Soo Park. Fast algorithms for projected clustering. In *Proc. of SIGMOD Conference*, 1999.

natural / output	Sky	Foliage	Cement	Window	Path	Grass
Outlier	0	6	0	5	2	6
0	0	21	4	10	5	0
1	0	0	0	0	0	0
2	30	3	4	9	2	0
3	0	0	22	6	16	0
4	0	0	0	0	5	24

(a)

natural / output	Sky	Foliage	Cement	Window	Path	Grass
Outlier	0	2	1	0	4	0
0	30	3	2	7	0	0
1	0	14	11	8	16	1
2	0	10	14	8	5	5
3	0	1	2	7	0	0
4	0	0	0	0	5	24

(b)

TABLE VII

CONFUSION MATRIX OBTAINED FOR THE REAL DATA SET (A)EPC2, (B)ORCLUS.

[5] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional space. In *Proc. of SIGMOD Conference*, 2000.

[6] R. Aggarwal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining application. In *Proc. of ACM SIGMOD Conference*, 1998.

[7] R. Aggrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of ACM SIGMOD Conference*, 1993.

[8] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of VLDB Conference*, 1994.

[9] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jorg Sander. Optics: ordering points to identify the clustering structure. In *Proceedings of the 1999 International Conference on Management of Data*, 1999.

[10] Chung-Hung Cheng, A. Fu, and Yi Zhang. Entropy-based subspace clustering for numerical data. In *Proc. of ACM SIGKDD Conference*, 1999.

[11] M. Datar E. Cohen, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J.D. Ullman, and C. Yang. Finding interesting associations without support pruning. In *Proc. of ICDE*, 2000.

[12] M. Ester, H-P. Kriegel, and X.Xu J. Sander. A density–based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of ACM SIGKDD Conference*, 1996.

[13] Brian Everitt. Cluster analysis, second edition. In *Halsted Heinemann*, 1980.

[14] A. Hinneburg and D.A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proc. of VLDB Conference*, 1999.

[15] Yannis E. Ioannidis and Viswanath Poosala. Balancing histogram optimality and practicality for query result size estimation. In *Proc. of ACM SIGMOD Conference*, 1990.

[16] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. In *ACM Computing Surveys, Vol. 31, No. 3*, 1999.

[17] Beyer K., Goldstein J., Ramakrishnan R., and Shaft U. When is nearest neighbor meaningful. In *Proc. of ICDT Conference*, 1999.

[18] D. Lin and Z.M. Kedemt. Pincer–search: A new algorithm for discovering the maximum frequent set. In *Proc. of Conference on Extending Database Technology (EDBT)*, 1998.

[19] Eric Ka Ka Ng and Ada Wai chee Fu. An effi cient algorithm for projected clustering. In *Proc. of ICDE Conference*, 2002.

[20] R. T. Ng and J. Han. Effi cient and effective clustering methods for spatial data mining. In *Proc. of VLDB Conference*, 1994.

[21] M. Procopiuc, M. Jones, P. K. Agarwal, and T.M. Murali. A monte carlo algorithm for fast projective clustering. In *Proc. ACM SIGMOD*, 2002.

[22] D. W. Scott. On optimal and data-based histograms. In *Biometrika, 66:605-610*, 1979.

[23] B. W. Silverman. Density estimation for statistics and data analysis. In *Chapman and Hall*, 1986.

[24] Scott D. W. Multivariate density estimation. In *Wiley & Sons*, 1992.

[25] W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the* 23^rd *VLDB Conference, pages* ***186–195***, 1997.

[26] T. Zhang, R.Ramakrishnan, and M. Livny. Birch: An effi cient data clustering method for very large databases. In *Proc. of ACM SIGMOD Conference*, 1996.

Eric Ka Ka Ng received the Bsc and MPhil degrees in computer science and engineering in the Chinese University of Hong Kong in 2000 and 2002 respectively. He is now working in a company expertizing in engineering Wireless LAN Management System, and utlizing database to provide consistent view of the managed Radio Network. His research interests include data mining, database, and wireless network.

Ada Wai-Chee Fu received the BSc degree in computer science in the Chinese University of Hong Kong, and both the MSc and the PhD degrees in computer science in Simon Fraser University of Canada. She worked at Bell Northern Research in Ottawa, Canada from 1989 to 1993 on a wide-area distributed database project; joined the Chinese University of Hong Kong in 1993, where she is an associate professor at the Department of Computer Science and Engineering. Her research interests include database related issues, data mining, parallel and distributed systems.

Raymond Chi-Wing Wong received the BSc and MPhil degrees in computer science and engineering in the Chinese University of Hong Kong in 2002 and 2004, respectively. He is now working as a research assistant in the Chinese University of Hong Kong. His research interests include data mining, database, security and video compression.