HARP: A Practical Projected Clustering Algorithm

Kevin Y. Yip, David W. Cheung, *Member, IEEE,* and Michael K. Ng

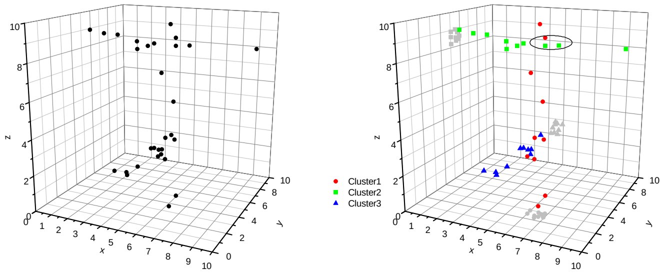
***Abstract*— In high-dimensional data, clusters can exist in subspaces that hide themselves from traditional clustering methods. A number of algorithms have been proposed to identify such projected clusters, but most of them rely on some user parameters to guide the clustering process. The clustering accuracy can be seriously degraded if incorrect values are used. Unfortunately, in real situations it is rarely possible for users to supply the parameter values accurately, which causes practical difficulties in applying these algorithms to real data. In this paper, we analyze the major challenges of projected clustering and suggest why these algorithms need to depend heavily on user parameters. Based on the analysis, we propose a new algorithm that exploits the clustering status to adjust the internal thresholds dynamically without the assistance of user parameters. According to the results of extensive experiments on real and synthetic data, the new method has excellent accuracy and usability. It outperformed the other algorithms even when correct parameter values were artificially supplied to them. The encouraging results suggest that projected clustering can be a practical tool for various kinds of real applications.**

***Index Terms*— Data mining, Mining methods and algorithms, Clustering, Bioinformatics.**

I. Introduction

Data mining is a process to discover unobserved object relationships. Clustering is one of the most well studied techniques, which concerns the partitioning of similar objects into clusters such that objects in the same cluster share some unique properties. Although some clustering algorithms have been proposed for thirty years [1], clustering remains a hot research topic and new algorithms emerge from time to time. This is mainly due to the ever-increasing variety, complexity and size of datasets. The need for faster and more specialized algorithms grows with the production of huge amount of data with diverse data characteristics.

In recent years, a special branch of clustering called projected clustering has been receiving a lot of attention from various communities. In projected clustering, clusters exist in subspaces of the input space defined by the dimensions[1] of the dataset. The similarity between different members of a cluster can only be recognized in the specific subspace. A dataset can contain a number of projected clusters, each forms in a possibly distinct subspace.



(a) A set of 3D objects. (b) 2-D projected clusters.

Fig. 1. An example of projected clusters.

A. Projected clusters

To illustrate the idea of projected clusters, consider the objects in Figure 1a. Although the distribution of objects suggests some underlying structures, it is hard to clearly define the clusters. The hidden relationships between the objects are revealed in Figure 1b, where the members of different clusters are given different shapes. By projecting the objects onto appropriate subspaces (see the shadows on the axis planes), the cluster structures become apparent. Should the corresponding subspaces of each cluster be not identified, the circled objects in Figure 1b would very likely be wrongly grouped into the same cluster due to their closeness in the 3D input space.

Projected clusters can appear in various kinds of data. Projected clustering has been successful in a computer vision task [2], and has potential applications in e-commerce [3]. We will also show in

Kevin Y. Yip and David W. Cheung are with the Department of Computer Science and Information Systems, University of Hong Kong, Hong Kong. E-mail: {ylyip, dcheung}@csis.hku.hk.

Michael K. Ng is with the Department of Mathematics, University of Hong Kong, Hong Kong. E-mail: mng@maths.hku.hk.

[1] In this paper, the terms “dimension” and “attribute” will be used interchangeably to mean the same concept.

Section IV that it outperforms traditional clustering methods on a gene expression dataset for cancer study and a food nutrition dataset.

For the sake of discussion, let us define a number of terms and notations. Given a dataset with N objects and a set V of d dimensions, a projected cluster C_i contains N_i member objects, and is defined in a subspace formed by the set V_i of d_i dimensions. d_i is referred to as the *dimensionality* of cluster C_i. As in most other studies (e.g. [2], [5]), we require each V_i to be a subset of V as the clustering results are easier to interpret. We will call the dimensions in V_i the *relevant dimensions* of C_i, and the ones in $V - V_i$ the *irrelevant dimensions* of it. The subspace formed by the two sets of dimensions will be called the *relevant subspace* and *irrelevant subspace* of C_i respectively. A dimension can be relevant to zero, one, or more clusters.

A dimension is relevant to a cluster if it helps distinguish the members of the cluster from other objects. In other words, in the relevant subspace of a cluster, the members of the cluster are similar to each other but dissimilar from other objects. In this paper we assume object similarity is measured by a distance metric, such as Euclidean distance. When all objects are projected onto a relevant dimension of a cluster, the projections of its members will be concentrated on a small range of values that contains few or no projections of other objects. The value ranges on the various relevant dimensions are called the "signature" of the cluster. For example, in Figure 1, a possible signature of cluster 1 is the axis-parallel rectangle on the x-y plane with extreme points $(6, 2)$ and $(7, 3)$. If the projection of an arbitrary object on the x-y plane falls into this region, it is likely to be a member of cluster 1. Notice that we cannot simply conclude that the object *is* a member of cluster 1 since in real datasets errors occur frequently. A cluster member may not abide by the signature of the cluster on some relevant dimensions (e.g. a member of cluster 1 may have the coordinates $(6, 8, 3)$ where the y-coordinate does not agree with the signature), or a non-member may have part of the signature by chance.

B. Projected clustering

The projected clustering problem is to identify a set of clusters *and their relevant dimensions* such that intra-cluster similarity is maximized while inter-cluster similarity is minimized. This is very similar to the traditional (non-projected) clustering problem [6], but is more general in that it allows each cluster to have only a subset of dimensions being relevant to. In this paper we assume clusters are disjoint, i.e., each object belongs either to one cluster or the set of outliers.

We assume that there is a set of *real clusters* that best matches the domain knowledge. The relevant dimensions of the real clusters are called the *real relevant dimensions* of them. Although the real clusters are rarely known to users, we will assume the existence of them for the sake of discussion. As clustering is an unsupervised-learning problem, all clustering algorithms studied in this paper do not make use of the information about real clusters.

In the remaining of this paper, the term *cluster* alone will mean a cluster produced by a clustering algorithm. The relevant dimensions of a cluster determined by an algorithm will be called its *selected dimensions* and the subspace formed by the dimensions the *selected subspace*. A cluster is *correct* if it contains objects all from the same real cluster, and *incorrect* otherwise.

In Figure 1, cluster 1 has a perfect signature along every relevant dimension in that no objects in other clusters are projected onto the signature range. Selecting a single relevant dimension (either x or y) for the cluster is enough to unambiguously identify all its members. In real datasets, due to the presence of errors, it is usually needed to select multiple relevant dimensions in order to identify all the members correctly. A clustering algorithm may assign a *relevance* value to each dimension of a cluster to indicate how well it helps identify the members of the cluster.

There are two major challenges in projected clustering that make it distinctive from the traditional clustering problem. The first challenge is the simultaneous determination of both cluster members and selected dimensions. Cluster members are determined by calculating object distances in the selected subspace, while the selected dimensions are determined by measuring the projected distances between cluster members. One common approach to tackling this chicken-and-egg problem is to form some tentative clusters according to some heuristics, determine their selected dimensions, and then refine the cluster members based on the selected dimensions. The heuristics being used are critical to the effectiveness of the algorithm. For instance, some existing algorithms make use of object distances in the input space to predict the members of a cluster, which could be quite inaccurate should the dimensionalities of the real clusters are small relative to the dataset dimensionality.

The second challenge is the evaluation of cluster quality, which is in turn related to the determination of the dimensionality of each cluster. Traditionally, objective functions are used to evaluate the quality of clusters. For example, k-means [1] assumes that each cluster is composed of objects distributed closely around the centroid. The objective of k-means is thus to minimize the average squared distance between each object and the centroid of its cluster. Some projected clustering algorithms [4], [5] generalize the function for projected clustering by considering only the selected dimensions in distance calculations. A weakness of this generalized function is that a better objective score can always be obtained by selecting fewer dimensions [7]. This could be a problem if the cluster signatures are not perfect, when each cluster will be tempted to select only one best dimension, which is not enough to identify all member objects correctly.

Some algorithms require users to supply the average cluster dimensionality as a constraint on the number of selected dimensions. This is a simple solution to the problem, but it in turn creates a usability problem as users are rarely able to supply the value accurately in real situations.

Another solution is to design a new objective function for projected clustering. Summarizing the proposals of some previous studies [2], [4], [5], a projected cluster is likely to be correct if

1) Its selected dimensions have high relevance.
2) It has a large number of selected dimensions.
3) It contains a large number of objects.

The reason for the first criterion is trivial, and the other two criteria ensure that the high relevance of the selected dimensions is not due to random chance [7]. It is favorable for a cluster to have all three properties, but in reality optimizing one property would usually sacrifice the other. Suppose a dimension is selected for a cluster if the average distance between the projected values is below a certain threshold, then when the threshold is fixed, adding more objects to a cluster will probably decrease the number of relevant dimensions qualified for selection. In the same manner, if the members of a cluster are fixed, raising the threshold will probably reduce the number of dimensions qualified for selection. Again, a simple way to deal with the problem is to combine the criteria into a single score, and let users to decide the relative importance of each criterion, which would also introduce a usability problem.

In summary, tentative clusters formation, clusters evaluation and the determination of cluster dimensionalities are the major difficulties of projected clustering. In the next section, we will study in more details some proposed projected clustering algorithms and discuss their potential weaknesses. In Section III we will introduce a new algorithm that 1) avoids the formation of incorrect clusters by allowing only the clusters with the highest chance of being correct to be formed, and 2) determines the dimensionality of clusters by dynamically adjusting its internal thresholds without relying on user inputs.

II. Related work

The partitional approach PROCLUS [5] is based on the k-medoids method [8]. As in traditional k-medoids methods, some objects are initially chosen as the medoids. But before assigning every object in the dataset to the nearest medoid, each medoid is first assigned a set of neighboring objects that are close to it in the input space to form a tentative cluster. For each tentative cluster, all dimensions are sorted according to the average distance between the projections of the medoid and the neighboring objects. On average l dimensions with the smallest average distances are selected as the relevant dimensions for each cluster, where l is a user parameter. Normal object assignment then resumes, but the distance between an object and a medoid is computed using only the selected dimensions. Medoids with too few assigned objects are replaced by some other objects to start a new iteration.

The user parameter l may introduce a usability problem since its correct value is hard to determine. Another potential problem arises when the real clusters have few relevant dimensions, in which case different members of a cluster may not be close to each other in the full input space. As a result, when a member of a real cluster is chosen as a medoid, the neighboring objects assigned to it may not come from the same real cluster. Subsequently, the dimensions selected would not be the real relevant dimensions and the resulting cluster would be mixed of objects from different real clusters.

Another partitional algorithm ORCLUS [4] was proposed to improve PROCLUS. According to the experimental results reported in [4], it is more accurate and stable than PROCLUS. Nevertheless, it still relies on user-supplied values in deciding the number of dimensions to select for each cluster.

In the hypercube approach DOC and its variant FastDOC [2], each cluster is defined as a hypercube with width 2ω, where ω is a user parameter. The clusters are formed one after another. To find a cluster, a pivot point is randomly chosen as the cluster center and a small set of objects is randomly sampled to form a tentative cluster around the pivot point. A dimension is selected if and only if the distance between the projected values of every sample and the pivot point on the dimension is no more than ω. The tentative cluster is thus bounded by a hypercube with width 2ω. All objects in the dataset falling into the hypercube are grouped to form a candidate cluster. More random samples and pivot points are then tried to form more candidate clusters, and a specially designed function is used to evaluate the quality of them. The candidate cluster with the best evaluation score is accepted, and the whole process repeats to find other clusters.

As with PROCLUS and ORCLUS, the selected dimensions of DOC and FastDOC are determined by a user parameter. In addition, they also restrict each cluster to be a hypercube with equal width along all relevant dimensions, which is unlikely to be true in real data. Tentative clusters are formed by random sampling, which avoids direct distance calculations in the input space. However, the number of tentative clusters required to try can become so large that seriously affects the speed performance.

There are some other proposed algorithms that determine object similarity based on the likeliness of the rise and fall patterns of projected values across the relevant dimensions instead of the absolute distance between the objects in the relevant subspace. We refer to this problem as *pattern-based*, as opposed to the *distance-based* clustering problem studied in this paper. Some pattern-based methods include pCluster [3] and MaPle [9]. The pattern-based approach has special values in some application domains such as bioinformatics and time-series data analysis. An interesting proposal for performing pattern-based clustering using a distance-based projected clustering algorithm can be found in [7].

There are also two computational problems closely related to projected clustering: subspace clustering [10] and biclustering [11]. The goal of the former problem is to search for all high-density regions in all subspaces (as opposed to returning only a small set of best clusters in projected clustering), and that of the latter is to search for possibly non-disjoint data submatrices that optimize certain (usually pattern-based) objective functions. Since the main focus of this paper is on projected clustering, we refer interested readers to the thorough survey of the three problems in [7].

In the next section, we will introduce a new algorithm HARP (a Hierarchical approach with Automatic Relevant dimension selection for Projected clustering), which is based on the traditional agglomerative hierarchical approach [12]. At the beginning each object is treated as a cluster, which are subsequently merged to form larger clusters. The other stream of hierarchical algorithms is the divisive methods, which put all objects into a single cluster at the beginning, and iteratively divide a cluster into smaller clusters. Some classical hierarchical algorithms can be found in [1], [12]. A brief introduction to some important concepts of hierarchical algorithms, such as linkage and object/cluster similarity can be found in [7]. Some recent developments of hierarchical clustering algorithms on handling irregular-shaped clusters and categorical attributes can be found in [13] and [14] respectively.

III. The new approach

A. Relevance index, cluster quality and merge score

We first define a function for measuring the relevance of a dimension to a cluster. In many previous studies [2], [4], [5], relevance is directly measured by the distance between projected values. This may not be appropriate if the input dimensions have different ranges of values. Consider an example relation shown in Table I, where objects 1 and 2 form a real cluster. If relevance is measured by the average within-cluster distance, dimension D is most relevant to the cluster as the within-cluster distance between projected values is smallest along the dimension. Similarly, if the measurement is based on average between-cluster distance, dimension C is most relevant to the cluster. Clearly, both proposals are problematic as they do not satisfy

TABLE I

AN EXAMPLE ILLUSTRATING THE IDEA OF RELEVANCE.

	Dim. A	Dim. B	Dim. C	Dim. D
Object 1	1	0.2	10	0.72
Object 2	2	0.3	30	0.70
Object 3	8	1.0	20	0.73
Object 4	9	0.9	40	0.71

the fundamental property of relevant dimensions: helping the distinguishing of cluster members. In comparison, dimensions A and B are actually more relevant to the cluster.

It is observed that for a dimension to be relevant to a cluster, not only should the projections of the cluster members be close to each other, the closeness should also be uncommon among the distance between the projections of any two objects. This can be captured by a comparison of the variance within the cluster (the *local variance*) and in the whole dataset (the *global variance*). Suppose the variance of projected values on dimension v_j in cluster C_i and in the whole dataset are σ_{ij}^2 and σ_j^2 respectively, the *relevance index* of v_j for C_i is defined as follows:

$$R_{ij} = 1 - \frac{\sigma_{ij}^2}{\sigma_j^2}. \tag{1}$$

The index gives a high value when the local variance is small compared to the global variance. This refers to the situation where the projections of the cluster members on the dimension are close, and the closeness is not due to a small average distance between the projected values in the whole dataset. A dimension receives an index value close to the maximum value (one) if the local variance is extremely small, which means the projections form an excellent signature for identifying the cluster members. Alternatively, if the local variance is only as large as the global variance, the dimension will receive an index value of zero. This suggests a baseline for dimension selection: a negative R value indicates a dimension is not more relevant to a cluster than to a random sample of objects. The dimension should therefore not be selected. We will discuss later how this baseline is used to define a stopping criterion of HARP.

To prevent the index from being undefined in some degenerating situations, we assume there does not exist any dimension with zero global variance. If such a dimension does exist, it is not useful at all and can be safely removed before the clustering process. Also, if a cluster contains only one object, the index values of all dimensions are set to one.

In Table I, the R values of the four dimensions for the cluster that contains objects 1 and 2 are 0.97, 0.97, -0.2 and -0.2 respectively, which match the intuitive relevance of the dimensions.

Based on the relevance index, the quality of a cluster C_i can be measured as the sum of the index values of all the selected dimensions:

$$Q_i = \sum_{v_j \in V_i} R_{ij}. \tag{2}$$

In general, the more selected dimensions a cluster has, and the larger are their respective R values, the larger will be the value of Q. We will discuss how HARP determines the relevant dimensions of each cluster later. At this point it can be assumed that each cluster has a reasonable set of selected dimensions.

Similarly, a score can be defined to evaluate the merge of two clusters. Basically, if two clusters can be merged to form a cluster with high quality, the merge is a potentially good one, i.e., the two clusters probably contain objects from the same real cluster. However, in case the two merging clusters have a large size difference, an unfavorable situation called *mutual disagreement* can occur. Consider a large cluster with a thousand objects and a small one with only five objects. If they are merged to form a new cluster, the mean and variance of projected values will highly resemble the original values of the large cluster, which will dominate the choice of the dimensions to be selected. If a dimension is originally selected by the large cluster, it will probably be selected by the new cluster also no matter the projected values of the small cluster are close to those of the large cluster or not. The resulting cluster can have a high Q score even the two clusters have a strong mutual disagreement on the signatures of the resulting cluster.

To cope with this problem, we modify the relevance index to take into account the mutual disagreement effect. Suppose C_{i_3} is the resulting cluster formed by merging C_{i_1} and C_{i_2}, the mutual-disagreement-sensitive relevance index of C_{i_3} on dimension v_j is defined as follows:

$$R^*_{i_3 j} \;=\; \frac{R_{i_1 j|i_2} + R_{i_2 j|i_1}}{2} \tag{3}$$

$$\begin{aligned} R_{i_1 j|i_2} &= 1 - \frac{\sigma^2_{i_1 j} + (\mu_{i_1 j} - \mu_{i_2 j})^2}{\sigma^2_j} \\ &= 1 - \frac{\sum_{x \in C_{i_1}} (x_j - \mu_{i_2 j})^2 / N_i}{\sigma^2_j}, \end{aligned} \quad (4)$$

$\mu_{i_1 j}$ and $\mu_{i_2 j}$ are the mean projected values on v_j of the two clusters respectively. The numerator of the second term of $R_{i_1 j|i_2}$ is the average squared distance between the projected values of C_{i_1} on v_j from the mean projected value of C_{i_2}. $R_{i_2 j|i_1}$ is defined similarly. The penalty term $(\mu_{i_1 j} - \mu_{i_2 j})^2$ ensures that the two clusters agree on the resulting signature. When mutual disagreement occurs, the penalty term will have a large value and the relevance index value will be attenuated. On the other hand, if the two mean values are equal, the relevance index will depend only on the local variance of the two clusters. The original R index is used to determine the quality of a cluster, while the modified index R^* is used to determine the merge score between two clusters. When C_{i_1} and C_{i_2} are merged to form C_{i_3}, the merge score is as follows:

$$\begin{aligned} & MS(C_{i_1}, C_{i_2}) \\ = & \sum_{v_j \in V_{i_3}} R^*_{i_3 j} \\ = & \sum_{v_j \in V_{i_3}} \frac{R_{i_1 j|i_2} + R_{i_2 j|i_1}}{2} \\ = & \sum_{v_j \in V_{i_3}} 1 - \frac{\sigma^2_{i_1 j} + \sigma^2_{i_2 j} + 2(\mu_{i_1 j} - \mu_{i_2 j})^2}{\sigma^2_j}. \end{aligned} \quad (5)$$

B. Validation of similarity scores

The MS function concerns both the quality and number of selected dimensions. As discussed in Section I-B, a third criterion for evaluating cluster quality is the cluster size. Suppose there is a set C of objects all belong to real clusters with dimension v_j being irrelevant to them. If the size of C is small, it is common to find the objects in C being close to each other along v_j by chance. If C is large, the chance for the same phenomenon to occur is small. Looking in another way, if a cluster has a high relevance index value at a dimension, the more objects the cluster contains, the less likely the high index value is merely by chance.

Since HARP is a hierarchical algorithm with each initial cluster containing a single object, it is not meaningful to incorporate cluster size into the merge score. However, it is possible to utilize the *potential* cluster size, which can be obtained from the distribution of projected values. Figure 2 shows the histogram built from such a distribution of a typical dimension that is relevant to some real clusters. The peaks correspond to the signatures of the real clusters. The base level at the troughs is likely due to random values. Suppose a cluster contains members with projected values within the interval $[a, b]$, it has a high potential to be merged with other clusters to become a cluster with a significant size and a high concentration of projected values around the $[a, b]$ region. On the other hand, if a cluster contains members with projected values within the interval $[c, d]$, although the cluster may receive a high R value at the dimension, the cluster is unable to keep the high R value if it is to grow to a significant size. The corresponding R value should therefore be invalidated in order to prevent more objects to be attracted to the cluster by the fake signature.

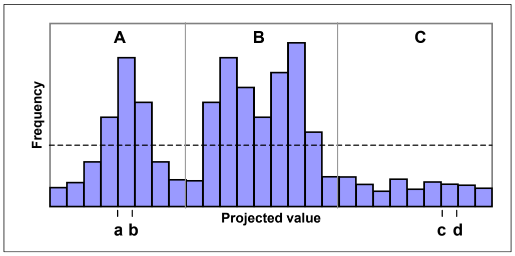


Fig. 2. A histogram built from the distribution of projected values of a typical dimension that is relevant to some real clusters.

Based on the observation, a histogram-based validation mechanism is developed to avoid the formation of incorrect clusters due to the above problem. The idea is that if a dimension is relevant to a cluster, the corresponding histogram should contain a peak around the signature values (see regions A and B in Figure 2). The width and height of the peak depend on the properties of the cluster, but provided the cluster has a significant size, the peak should exceed the random noise level, which corresponds to the mean frequency in case of a uniform distribution (shown by the dotted line). Clusters covered by bins that stay below the noise level are statistically insignificant (region C), and their relevance index values for the dimension will be rejected.

The validation mechanism contains two steps. First, the Kolmogorov-Smirnov goodness of fit test [15] is used to remove dimensions that are likely irrelevant to all clusters, i.e., the dimensions whose distributions are essentially uniform. Each remaining dimension is expected to be relevant to at least one cluster. If a cluster C_i has mean μ_{ij} and variance σ_{ij}^2 at dimension v_j, we check the mean frequency of the bins covering the range $[max(\mu_{ij} - 2\sigma_{ij}, min_{ij}), min(\mu_{ij} + 2\sigma_{ij}, max_{ij})]$, where min_{ij} and max_{ij} are the minimum and maximum projected values of the members of C_i on v_j. The use of a 4-standard deviation range covers 95% of the projected values if they follow a Gaussian distribution and ignore some abnormalities, while min_{ij} and max_{ij} refine the boundaries of the range for non-Gaussian cases. When selecting the relevant dimensions of a cluster, if the mean frequency of the bins is below the mean of all the bins, R_{ij} will be set to zero. When calculating MS between two clusters C_{i_1} and C_{i_2}, if either R_{i_1j} or R_{i_2j} is set to zero by the validation mechanism, v_j will make a zero contribution to the MS score. An empirical evaluation of the effectiveness of the validation mechanism will be given in Section IV-D.2.

C. Dynamic threshold loosening

When we introduced the MS function in Section III-A we assumed that there is a way to determine the relevant dimensions of each cluster. In this section we discuss how it is made possible by the dynamic threshold loosening mechanism.

As discussed before, a cluster is likely to be correct if it contains a large number of selected dimensions, and the selected dimensions have high relevance index values. This means merges that form resulting clusters with both properties should be performed earlier. This is achieved by two internal thresholds R_{min} and d_{min}. Two clusters are allowed to be merged if and only if the resulting cluster has d_{min} or more selected dimensions, and a dimension v_j is selected for a potential cluster C_i if and only if $R^*_{ij} \geq R_{min}$. At any time, the two thresholds define a set of allowed merges where the actual merging order within the set is determined by the MS scores.

At the beginning, R_{min} and d_{min} are initialized to the tightest (i.e., highest) values 1 and d (dataset dimensionality) respectively. All allowed merges produce clusters that contain identical objects, which must be correct. Whenever all qualified merges have been performed, the thresholds will be slightly loosened. As clustering proceeds, the clusters grow bigger in size. The projections of the cluster members on the real relevant dimensions remain close to each other, but the chance of having similar closeness of projections on other dimensions drops, so as their relevance index values. This allows the real relevant dimensions to be differentiated from the irrelevant ones, which in turn ensures the formation of correct clusters.

In order to guarantee the quality of clusters, the two thresholds are associated with baseline values such that when the baselines are reached, no further loosening is allowed. As mentioned in Section III-A, a negative R value means that a dimension is very unlikely to be relevant to a cluster. The baseline of R_{min} is thus set to zero. For d_{min}, the baseline is set to one, which is the minimum value for a cluster to be defined as a projected cluster. We will see later that the HARP algorithm allows users to specify an optional target number of clusters. According to our experience, if such a value is specified, the algorithm usually finishes the clustering process well before the thresholds reach their baselines. The clusters produced thus contain selected dimensions with R values much better than that of a random set of projected values.

There are many possible ways to loosen the threshold values. From our empirical study, a simple linear loosening scheme is found to be very adaptive and performed well. In this scheme, there is a fixed number of threshold levels such that whenever no more qualified merges remain, the values of the two thresholds are updated using a linear interpolation towards the baseline values (see Section III-D for details). By default, we set the number of threshold levels to the dataset dimensionality d such that after each loosening, d_{min} is reduced by 1. In general, using a larger number of levels will lead to a better accuracy but a longer execution time. We will show in Section IV that the clustering accuracy is insensitive to small changes to the number of levels.

D. The algorithm

With the core building blocks explained, we now present the complete algorithm, which is described by the pseudo codes in Table II.

At the beginning of the clustering process, each object is a separate cluster. The two thresholds d_{min} and R_{min} are set to their tightest values. For each cluster, the dimensions that satisfy the threshold

TABLE II

THE HARP ALGORITHM.

```
Algorithm HARP (k: target no. of clusters (default: 1))
1   For $step$ := 0 to $d-1$ do {
2       $d_{min}$ := $d - step$
3       $R_{min}$ := $1 - step/(d-1)$
4       Foreach cluster $C_i$
5           SelectDim($C_i$, $R_{min}$)
6       BuildScoreCache($d_{min}$, $R_{min}$)
7       While cache is not empty {
8           // $C_{i_1}$ and $C_{i_2}$ are the clusters involved in the
9           // best merge, which forms the new cluster $C_{i_3}$
10          $C_{i_3}$ := $C_{i_1} \cup C_{i_2}$
11          SelectDimNew($C_{i_3}$, $R_{min}$)
12          UpdateScoreCache($C_{i_3}$, $d_{min}$, $R_{min}$)
13          If clusters remained = $k$
14              Goto 17
15      }
16  }
17  ReassignObjects()
End
```

requirements are selected. The merge score between each pair of clusters is then calculated. All merges that form a resulting cluster with less than d_{min} selected dimensions are not allowed to perform.

The algorithm repeatedly performs the best merge according to the merge scores of the qualified merges. In order to efficiently determine the next best merge, merge scores are stored in a cache. After each merge, the scores related to the merged clusters are removed from the cache, and the best scores of the qualified merges that involve the new cluster are inserted. The selected dimensions of the new cluster are determined by its members according to R_{min}. Due to the definition of R, if a dimension is originally not selected by both merged clusters, it must not be selected by the new cluster. But if a dimension is originally selected by one or both of the merging clusters, it may or may not be selected by the new cluster.

Whenever the cache becomes empty, no more qualified merges exist at the current threshold level. The thresholds will be loosened linearly as explained before (lines 2-3 of Table II). Further rounds of merging and threshold loosening are carried out until a target number of clusters remain, or the thresholds reach their baseline values and no more merging is possible.

To further improve clustering accuracy, an optional object reassignment step can be performed after the completion of the hierarchical part. The MS score between each clustered object and each cluster is computed based on the final threshold values when the hierarchical part ends. After computing all the scores, each of the objects is assigned to the cluster with the highest MS score. The process repeats until convergence or a maximum number of iterations is reached.

Finally, we describe the outlier handling mechanism of HARP. It is similar to the one used by CURE [13] with two phases of outlier removal. Phase one is performed when the number of clusters remained reaches a fraction of the dataset size. Clusters with very few objects are removed. Phase two is performed near the end of clustering to prevent the merge of different real clusters due to the existence of noise clusters. As pointed out in [13], the time to perform phase one outlier removal is critical. Performing too early may remove many non-outlier objects, while performing too late may have some outliers already merged into clusters. HARP performs phase one relatively earlier so that most outliers are removed, possibly together with some other objects. Before phase two starts, each removed object is filled back to the most similar cluster subject to the current threshold requirements. Due to the thresholds, real outliers are unlikely to be filled back. From experimental results, the fill back process usually improves the accuracy.

E. Complexity analysis

Suppose each cache access (insertion or deletion of all the merge scores that involve a cluster) takes $O(f(N))$ time, it can be shown that the whole algorithm takes $O(N^2d^2 + Nf(N))$ time [7]. We implemented three kinds of cache structures: priority queue (similar to the one used in [14]), quad tree and Conga line [16], with $f(N)$ ranges from N to $N \log^2 N$.

There are many ways to improve the speed performance of HARP. For two clusters to be qualified for merging, the number of common dimensions that pass the histogram-based validation must exceed d_{min}. By checking the maximum number of such common dimensions of all cluster pairs, many threshold levels can be skipped if they contain no possible merges. This optimization is most useful when the dimensionalities of the clusters are low relative to the dataset dimensionality. Similarly, when determining the merge score between two clusters,

TABLE III
DATA PARAMETERS OF THE SYNTHETIC DATASETS.

Parameter	Default Values
Dataset size (N)	500
Dataset dimensionality (d)	20
Number of clusters (k)	5
Cluster size (N_i)	15 to 25% of N
Average cluster dimensionality (l_{real})	$\frac{d}{k}$ to $0.9d$
Domain of dimensions ($[min_j, max_j]$)	[0,1] to [0,10]
Local S.D. of relevant dimensions (σ_{ij})	3 to 5% of domain
Artificial data errors (e)	5%
Artificial outliers (o)	0%

the R^* value of each dimension of the resulting cluster is computed in turn. Once the number of selected dimensions is confirmed to be lower than d_{min}, the R^* values of the remaining dimensions do not need to be computed.

When the dataset size is very large, it is also possible to perform clustering on a random sample only. Upon completion, each unsampled object is filled back to the most similar cluster subject to the restriction of the final threshold values. Similarly, when the dataset dimensionality is very high, a constant number of threshold levels can be used (line 2 of Table II), so that the quadratic term with respect to d in the total time complexity becomes linear. We will show in the next section that these speedup methods are feasible in practice.

The space complexity of HARP is $O(n)$ when Conga line is used, and $O(n^2)$ when quad tree or priority queue is used. Depending on the memory available, HARP chooses the best cache structure to use but produces identical clustering results.

IV. EXPERIMENTS

In this section we report various experimental results of HARP and some other clustering algorithms in comparison. More results can be found in [7].

A. Datasets

1) Synthetic data: Table III lists the default parameters used in synthetic data generation.

When generating a dataset, the size of each cluster and the domain of each dimension were first determined randomly according to the data parameters. Having different cluster sizes creates different peak heights at the distributions, which test the stability of the histogram-based validation mechanism. The different domain sizes are to test the effectiveness of the relevance index. Each cluster then randomly picked its relevant dimensions, where a single dimension could be relevant to multiple clusters. Since dimensions that are irrelevant to all clusters can be removed by feature selection techniques, which are not the major concern of the current paper, we made each dimension to be relevant to at least one cluster.

For each relevant dimension of a cluster, the local mean and standard deviation were chosen randomly from the domain to construct a Gaussian distribution. Each object in the cluster determined whether to follow the signature according to the data error rate e. This was to simulate experimental and measurement errors. If an object was chosen to follow the signature, a projected value would be sampled from the Gaussian distribution. Otherwise, and for all irrelevant dimensions, the values would be sampled from a uniform distribution over the whole domain.

2) Real data: Lymphoma: It is a gene expression dataset used in studying distinct types of diffuse large B-cell lymphoma (Figure 1 of [17]). It contains 96 samples, each with 4026 expression values of genes. The samples are categorized into 9 classes according to the category of mRNA sample studied. We clustered the samples with the genes as the input dimensions, and used the class labels to evaluate the clustering performance. Each selected dimension of a cluster represents a gene that has similar expressions in the member samples. If a cluster contains mainly one type of tumor samples, the genes are the potential signatures for identifying the presence of the specific type of tumor.

Food: It contains the weight and 6 attributes (Fat, Food Energy, Carbohydrate, Protein, Cholesterol and Saturated Fat) of 961 food items[2]. We followed [18] to divide the attribute values by the weight, and standardize each column to have unit standard deviation. Since the dataset contains no class labels, we treat the clustering as an exploratory task and report some interesting findings.

B. Comparing algorithms

To demonstrate the capability of HARP, we compared it with various projected and non-projected

[2] We downloaded the dataset from `http://www.ntwrks.com/~mikev/chart1.html`.

algorithms. For the projected side, we chose PROCLUS [5], ORCLUS [4] and FastDOC [2] as they have reasonable worst case time and are able to produce disjoint clusters, which makes it easy to compare the clustering results. FastDOC creates clusters one at a time. We used it to produce disjoint clusters by removing the clustered objects before forming a new cluster. After forming the target number of clusters, the unclustered objects were treated as outliers. For the non-projected camp, we chose a simple agglomerative hierarchical algorithm, two partitional algorithms CLARANS [8] and KPrototype [19] (based on k-medoids and k-means respectively), and CAST [20], a popular algorithm for clustering high-dimensional gene expression profiles. We believe our choice of algorithms covers a wide spectrum of clustering approaches.

C. Other details

1) Algorithm parameters: In all experiments the target number of clusters was set to the number of real clusters. For each of the other parameters, various reasonable values were tried (details can be found in [7]). CAST and FastDOC produced the desired number of clusters only at some specific parameter values. All results that form fewer than the desired number of clusters were discarded.

2) Execution: Each experiment was repeated five times to avoid bias due to randomness (e.g. locations of initial medoids). For each repeated run, only the result that has the best algorithm-specific objective score will be considered in the discussions below.

3) Evaluation criteria: We used the Adjusted Rand Index (ARI) [21] as the performance metric for clustering accuracy. It is based on the Rand Index [22], with the expected index value also taken into account. It measures how similar are the partition of objects according to the real clusters (U) and the partition in a clustering result (V). Denote a, b, c and d as the number of object pairs that are in the same cluster in both U and V, in the same cluster in U but not V, in the same cluster in V but not U, and in different clusters in both U and V respectively, ARI is defined as follows:

$$ARI(U, V) = \frac{2(ad - bc)}{(a+b)(b+d) + (a+c)(c+d)} \quad (6)$$

The more similar are the two partitions (larger a and d, smaller b and c), the larger will be the ARI value. When U and V are identical, the index value will be one. When V is only as good as a random partition, the index value will be zero.

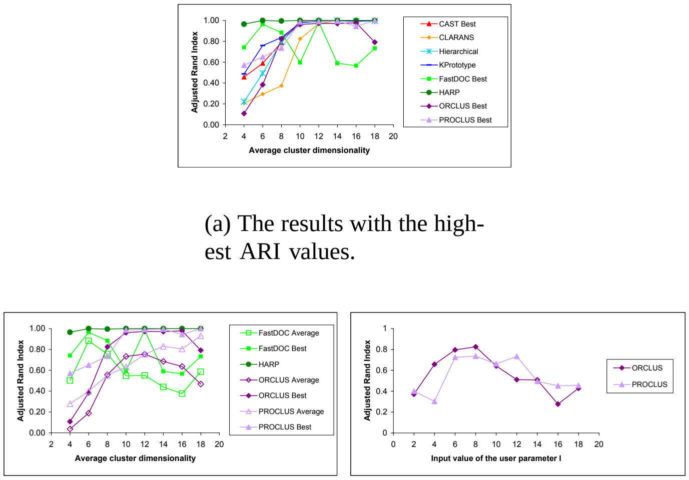
We used precision and recall to evaluate how similar are the selected dimensions and the real relevant dimensions. For each cluster, precision is the number of real relevant dimensions being selected divided by the number of selected dimensions. Recall is the number of real relevant dimensions being selected divided by the actual number of real relevant dimensions. The reported value of a clustering result is the average of all the clusters.

4) Data preprocessing: We generated an "easy-to-cluster" dataset with l_{real}=12 and o=0 to test the importance of data preprocessing. We tested the clustering accuracy of the projected algorithms with and without standardizing the values of each dimension, using correct user parameter values. The results (see [7] for details) show that with the global variance taking into account in the relevance index, the performance of HARP is invariant to the standardization process. For all the other methods, the clustering accuracy was improved by standardization. For fair comparisons, all the synthetic datasets used in the coming sections were standardized.

5) Outlier handling: From some preliminary experiments, we noticed that FastDOC and PROCLUS tend to discard a large amount of outliers even the dataset contains no or few artificial outliers. In order to give a fair comparison of the clustering results, except otherwise specified, the synthetic datasets used in the coming experiments contain no artificial outliers, and the outlier removal options of all algorithms were disabled. For CAST and FastDOC, the unclustered objects were still discarded as outliers, and we accept only results with discarding rates not more than 40%. To show the noise-immunity of HARP, there will be a separate subsection dedicated to experiments on noisy data.

D. Results on synthetic data

1) Clustering accuracy: The first set of experiments concerns how the clustering accuracy is affected by cluster dimensionality l_{real}. We generated eight datasets with l_{real} ranging from 4 to 18. For clarity, we present the results in three different charts in Figure 3. In the charts, and in the other figures to be presented later, a line labeled "best" and "average" represents the result with the highest

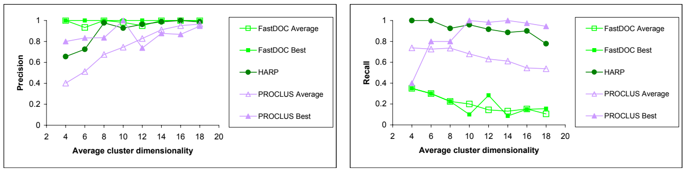


(a) The results with the highest ARI values.

(b) Comparing the results with the highest ARI values with the average results using different parameter values.

(c) Clustering accuracy of PROCLUS and ORCLUS when l_{real}=8, with various user parameter inputs.

Fig. 3. Clustering accuracy with different cluster dimensionalities.



(a) Precision of the selected dimensions.

(b) Recall of the selected dimensions.

Fig. 4. Accuracy of the selected dimensions.

ARI values and the average result after trying all the parameter values respectively. Since CLARANS, HARP, Hierarchical and KPrototype used only one set of parameter values, only one line is presented for each of them.

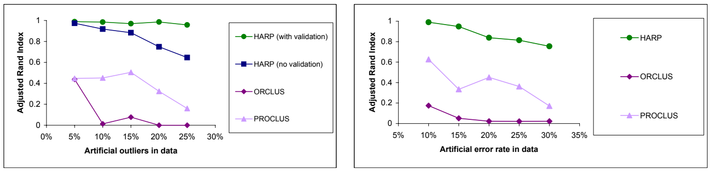
Figure 3a shows the best results (with the highest ARI values) of the algorithms. Most algorithms were highly accurate at large l_{real} values, but for l_{real} values lower than 50% of d, the performance difference between different algorithms became apparent. HARP got the highest ARI values among all algorithms on all datasets, and remained extremely accurate even each cluster had 80% of the dimensions irrelevant to them. The results of ORCLUS reported in [4] are better than the results observed in our experiments, which is likely caused by the small sizes of our synthetic datasets. ORCLUS works best on large datasets that contain sufficient values for performing PCA. In comparison, its performance on small datasets is less competitive. FastDOC continued to discard a large amount of non-outlier objects, with an average discarding rate of 26.3%, which equals the size of one to two complete clusters.

In general the projected algorithms outperformed the non-projected ones at small l_{real} values, but some good results were due to the correct input of parameter values. Figure 3b compares the best results of FastDOC, ORCLUS and PROCLUS when correct parameter values were used with their average results when a set of linearly chosen parameter inputs were used. The average results have much lower ARI values than the best results, which means in reality if the correct parameter values are unknown, the optimal results can hardly be obtained. Figure 3c shows the typical fluctuation of accuracy of PROCLUS and ORCLUS with various parameter inputs, taken from the results on the dataset with l_{real}=8. Both algorithms achieved their peak performance when correct inputs were supplied, but the error rates raised as the inputs moved away from the correct values. In comparison, the accuracy of HARP is independent of user inputs.

From Figure 3b, it is also noted that PROCLUS and ORCLUS did not perform well when l_{real} is small even correct parameter values were used. This is due to the formation of incorrect tentative clusters caused by object assignments that depend on distance calculations in the input space. In contrast, by allowing only merges with maximum number of selected dimensions, HARP was able to prevent from forming incorrect tentative clusters.

Next we investigate the selected dimensions of the projected clustering algorithms. Figures 4a and 4b show the average precision and recall values of the selected dimensions of the results produced by FastDOC, HARP and PROCLUS.

When l_{real} is large, HARP tended to be conservative in dimension selection as reflected by the high precision and relatively low recall values. This means HARP deliberately avoided selecting irrelevant dimensions when the selected ones were enough for identifying the cluster members correctly. However, when l_{real} is small, HARP tried to include all relevant dimensions in order not to miss any useful information, with the expense of also selecting some irrelevant dimensions. This is not a serious problem as including a few irrele-



(a) Clustering accuracy with the presence of outliers.

(b) Clustering accuracy with the presence of errors.

Fig. 5. Clustering results on imperfect data.

vant dimensions has only a moderate effect to the clustering accuracy if the signatures at the relevant dimensions are clear enough to identify the cluster members, while missing a single relevant dimension may mean missing a substantial proportion of information. If the accuracy of selected dimensions is critical to an application, a post-processing step can be carried out to rank all the dimensions of each cluster based on the R values, and filter out the unwanted dimensions according to the application-specific needs. Our argument is supported by the excellent accuracy of HARP at all l_{real} values.

The best results of FastDOC are characterized by excellent precision and fair recall values over the whole range of l_{real} values. This means it tends to be parsimonious in dimension selection, which can be a great problem when l_{real} is small. The behavior of the best results of PROCLUS is similar to HARP, but is relatively less stable. On the other hand, as expected, the average results of PROCLUS are not satisfactory except at very large l_{real} values.

2) Imperfect datasets: Although the above experiments show that HARP is highly accurate, the datasets being used are too ideal with no outliers, low error rates and clear signatures. In the coming experiments we demonstrate the influence of these data parameters on the clustering accuracy. We fixed l_{real} to 6 (30% of d) and generated three sets of data with increasing o, e and σ_{ij} respectively. We tested the performance of HARP, using PROCLUS and ORCLUS (with correct parameter values) as reference. The results are shown in Figure 5.

Figures 5a shows the results on the datasets with artificial outliers. From the figure, the performance of HARP was less sensitive to outliers than the other two algorithms, and the histogram-based validation mechanism was effective in improving the accuracy of HARP. In comparison, ORCLUS and PROCLUS had unsatisfactory performance. ORCLUS appears to be very sensitive to outliers, which may due to the fact that in late iterations it uses the centroids (instead of medoids) as cluster seeds. When the clustering accuracy is low, each cluster consists of objects from many different real clusters and the centroids will contain a mixture of their signatures. As a result, the centroids will be similar to each other, but dissimilar to any data objects, which ruin the outlier removal mechanism of ORCLUS.

Figure 5b shows the results with increasing amount of data errors. It shows that the accuracies of all three algorithms went down as more errors were introduced, but HARP only had a mild deterioration. Similar results are observed when the cluster signatures became less concentrated (figures can be found in [7]).

E. Scalability experiments

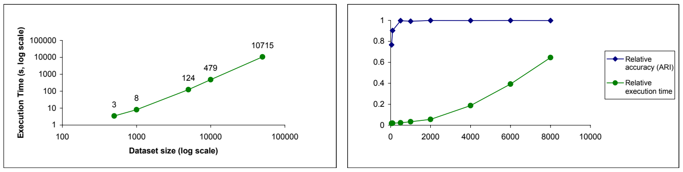
In this section we study the scalability of HARP with increasing dataset size and dimensionality. We tested the performance of HARP on two sets of data, the first with N increasing from 1000 to 500000 (using Conga line as cache structure), and the second with d increasing from 100 to 500 and average cluster dimensionality kept at 30% of d.

The results with increasing dataset size are shown in Figure 6a, which confirms that the actual execution time was bounded by the theoretic time complexity. For medium-sized datasets ($N \approx 10000$), the execution time was usually better than ORCLUS and FastDOC, and comparable to PROCLUS when the time used in repeated runs is also included. Part b shows the relative execution time and accuracy when the sample-based speedup technique described in Section III-E was applied to the dataset with 10000 objects. For reasonable sample sizes, the execution time was much improved with only a little impact on the accuracy.

The results with increasing dataset dimensionality are similar (figures can be found in [7]). The execution time was shown to be sub-quadratic with respect to d. When HARP was speeded up by using fewer threshold levels, the execution time was greatly reduced, but the clustering accuracy remained excellent.

F. Results on real data

For the lymphoma data, we used HARP and PROCLUS as the representatives of projected clus-



(a) Execution time with increasing N.

(b) Relative accuracy and execution time with various sample sizes.

Fig. 6. Clustering results of HARP with increasing N.

tering algorithms. HARP got an ARI value of 0.75, which is higher than the values obtained by all other algorithms. The samples of different types were well separated into different clusters, and most samples (43 out of 46) of the major class (DLBCL) were put into a single cluster. The clusters of HARP have 2014 to 3515 selected dimensions, corresponding to 50% to 87% of all dimensions. The selected dimensions display some biological significance. In Figure 2 of [17], some genes are highlighted as the signatures of some sample types or biological process. We ranked the selected genes of each cluster according to their R values, and found that a large number of relevant signature genes were selected by the clusters with very high ranking and R values. For example, a subcluster of the large DLBCL cluster has all signature genes in the proliferation region selected. Among the 3347 selected genes of it, all the signature genes in the region are within the top 700 ranked genes. About 70% of them are even within the top 75, with R values above 0.83.

For the food data, we used HARP to produce twenty clusters. Some interesting clusters can be found in [7]. For example, one of them contains all twelve margarine items in the dataset. Three of the dimensions have high relevance index values and were selected by HARP. However, the index values of the other three dimensions are low and they were therefore not selected. This means the margarine items are close in the selected three-dimensional subspace, but may not be close in the input space. We verified this by performing ten rounds of KPrototype on the data. In all cases, the twelve items were distributed to two or more clusters, which suggests that the non-projected clustering algorithm is unable to produce the same interesting cluster.

V. Discussions and Future Work

The dynamic threshold loosening mechanism of HARP is shown to be successful in eliminating the reliance on user parameters. We believe the concept of dynamic parameter tuning has a great potential value in problems where the algorithms usually rely on user parameters.

The experimental results also reveal that projected clustering is meaningful only when the dimensionalities of the clusters are well below the dataset dimensionality. We recommend further studies on projected clustering to focus on datasets with l_{real} not more than 30% of d. In some gene expression datasets, the number of relevant genes of each function group can be lower than 10% of the total number of genes. Most projected clustering algorithms (including HARP) may not perform well, while the subspace clustering approaches (such as [10]) introduced in Section II may run indefinitely long since the absolute value of l_{real} can be very high (e.g. 100). Further improvements of projected clustering algorithms are called for.

It is also interesting to see if HARP can be modified to produce pattern-based and non-disjoint clusters, which are more appropriate in some situations. A preliminary study can be found in [7].

VI. Conclusions

In this paper, we analyzed the major challenges of the projected clustering problem, and proposed a new algorithm HARP that does not depend on user inputs in determining the relevant dimensions of clusters. It makes use of the relevance index, histogram-based validation and dynamic threshold loosening to adaptively adjust the merging requirements according to the clustering status. Experimental results on synthetic and real data suggest that HARP has a higher accuracy and usability than the projected and non-projected algorithms being compared, and it remains highly accurate when handling noisy data. The interesting clusters discovered in the lymphoma and food data suggest that HARP could be a practical tool for real applications.

Acknowledgements

The research of the second author is supported by a grant from the Research Grant Council of Hong Kong. (Project number : HKU 7141/03E).

REFERENCES

[1] J. Hartigan, *Clustering Algorithms*. Wiley, 1975.

[2] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali, "A monte carlo algorithm for fast projective clustering," in *ACM SIGMOD International Conference on Management of Data*, 2002.

[3] H. Wang, W. Wang, J. Yang, and P. S. Yu, "Clustering by pattern similarity in large data sets," in *ACM SIGMOD International Conference on Management of Data*, 2002.

[4] C. C. Aggarwal and P. S. Yu, "Finding generalized projected clusters in high dimensional spaces," in *ACM SIGMOD International Conference on Management of Data*, 2000.

[5] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park, "Fast algorithms for projected clustering," in *ACM SIGMOD International Conference on Management of Data*, 1999.

[6] J. Han and M. Kamber, *Dara Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[7] K. Y. L. Yip, "HARP: A practical projected clustering algorithm for mining gene expression data," Master's thesis, The University of Hong Kong, Pokfulam Road, Hong Kong, 2004.

[8] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, 1994.

[9] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu, "MaPle: A fast algorithm for maximal pattern-based clustering," in *IEEE International Conference on Data Mining*, 2003.

[10] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *ACM SIGMOD International Conference on Management of Data*, 1998.

[11] Y. Cheng and G. M. Church, "Biclustering of expression data," in *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 2000.

[12] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Inter-Science, 1990.

[13] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," in *ACM SIGMOD International Conference on Management of Data*, 1998.

[14] ——, "ROCK: A robust clustering algorithm for categorical attributes," in *15th International Conference on Data Engineering*, 1999.

[15] P. Bickel and K. Doksum, *Mathematical Statistics, Basic Ideas and Selected Topics*. Holden-Day, Inc., Oakland, 1977.

[16] D. Eppstein, "Fast hierarchical clustering and other applications of dynamic closest pairs," in *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[17] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson, L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D.Weisenburger, J. O. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. C. Byrd, D. Botstein, P. O. Brown, and L. M. Staudt, "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, pp. 503–511, 2000.

[18] L. Lazzeroni and A. Owen, "Plaid models for gene expression data," *Statistica Sinica*, vol. 12, pp. 61–86, 2002.

[19] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in *The First Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 1997.

[20] A. Ben-Dor and Z. Yakhini, "Clustering gene expression patterns," in *Proceedings of the Annual International Conference on Computational Molecular Biology*, 1999.

[21] K. Yeung and W. Ruzzo, "An empirical study on principal component analysis for clustering gene expression data," *Bioinformatics*, vol. 17, no. 9, pp. 763–774, 2001.

[22] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, pp. 846–850, 1971.

Kevin Yuk-Lap Yip received his M.Phil degree in Computer Science in 2003. He is now a research assistant in the Department of Computer Science and Information Systems at the University of Hong Kong. His research interests include data mining, bioinformatics, and XML-based distributed systems.

David Wai-lok Cheung received the M.Sc. and Ph.D. degrees in computer science from Simon Fraser University, Canada, in 1985 and 1989, respectively. He also received the B.Sc. degree in mathematics from the Chinese University of Hong Kong. From 1989 to 1993, he was a Member of Scientific Staff at Bell Northern Research, Canada. Since 1994, he has been a faculty member of the Department of Computer Science in The University of Hong Kong. He is also the Director of the Center for E-Commerce Infrastructure Development. His research interests include data mining, data warehousing, XML technology for e-commerce and bioinformatics. Dr. Cheung was the Program Committee Chairman of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining and has served on the program committees in many database conferences. He is a member of the ACM and the IEEE Computer Society.

Michael Kwok-Po Ng is an Associate Professor in the Department of Mathematics and is also Adjunct Research Fellow in the E-Business Technology Institute at the University of Hong Kong. He did his undergraduate and M.Phil. degrees (from 1987 to 1992) at the University of Hong Kong and obtained his Ph.D. degree from Department of Mathematics at the Chinese University of Hong Kong in 1995. In 2001, he was selected as one of the recipients of the Outstanding Young Researcher Award of the University of Hong Kong. Michael's research areas are Data Mining, Operations Research and Scientific Computing. He has published more than 150 papers in the refereed journals and proceedings. He serves on the editorial boards of Numerical Mathematics, Multidimensional Systems and Signal Processing, and International Journal of Computational Science and Engineering. He is also the guest editor of the special issues in Numerical Linear Algebra with Applications, International Journal of Applied Mathematics, International Journal of Imaging Systems and Technology, and Applied Mathematics and Computation.