A Highly-usable Projected Clustering Algorithm for Gene Expression Profiles

Kevin Y. Yip
Department of Computer Science and Information Systems
University of Hong Kong
Pokfulam Road, Hong Kong
ylyip@csis.hku.hk

David W. Cheung*
Department of Computer Science and Information Systems
University of Hong Kong
Pokfulam Road, Hong Kong
dcheung@csis.hku.hk

Michael K. Ng
Department of Mathematics
University of Hong Kong
Pokfulam Road, Hong Kong
mng@maths.hku.hk

ABSTRACT

Projected clustering has become a hot research topic due to its ability to cluster high-dimensional data. However, most existing projected clustering algorithms depend on some critical user parameters in determining the relevant attributes of each cluster. In case wrong parameter values are used, the clustering performance will be seriously degraded. Unfortunately, correct parameter values are rarely known in real datasets. In this paper, we propose a projected clustering algorithm that does not depend on user inputs in determining relevant attributes. It responds to the clustering status and adjusts the internal thresholds dynamically. From experimental results, our algorithm shows a much higher usability than the other projected clustering algorithms used in our comparison study. It also works well with a gene expression dataset for studying lymphoma. The high usability of the algorithm and the encouraging results suggest that projected clustering can be a practical tool for analyzing gene expression profiles.

1. INTRODUCTION

Clustering is a popular data mining technique for various applications. One of the reasons for its popularity is the ability to work on datasets with minimum or no a priori knowledge. This makes clustering practical for real world applications. Recently, high dimensional data has aroused the interest of database researchers due to its new challenges brought to the community. In high dimensional space, the distance from a record to its nearest neighbor can approach its distance to the farthest record [6]. In the context of clustering, the problem causes the distance between two records of the same cluster to approach the distance between two records of different clusters. Traditional clustering methods may fail to identify the correct clusters.

In high dimensional datasets, clusters can be formed in subspaces. Only a subset of attributes is relevant to each cluster, and each cluster can have a different set of relevant attributes. An attribute is relevant to a cluster if it helps identify the member records of it. This means the values at the relevant attributes are distributed around some specific values in the cluster, while the records of other clusters are less likely to have such values. Finding clusters and their relevant attributes from a dataset is known as the projected (subspace) clustering problem. For each cluster, a projected clustering algorithm determines a set of attributes that it assumes to be most relevant to the cluster. We will call such attributes the "selected attributes" of the cluster.

Projected clustering is potentially useful in analyzing gene expression profiles. In these datasets, the transcript levels of many genes in different samples are recorded. We can view the expression levels of different genes as attributes of the samples, or the samples as the attributes of different genes. Clustering can be performed on genes or samples [7]. A set of related genes may coexpress simultaneously in only some samples. Alternatively, a set of related samples may have only some of the genes coexpressed simultaneously. Identifying the relevant attributes may help improve the clustering accuracy. The selected attributes may also suggest a smaller set of genes/samples for researchers to focus on, possibly reduces the efforts spent on expensive biological experiments.

As in the case of traditional clustering, the goal of projected clustering algorithms is to form clusters with optimal quality. However, the traditional functions used in evaluating cluster quality may not be applicable in the projected case. For example, if the average within-cluster distance to centroid is used within the selected subspace, the fewer attributes being selected, the better evaluation score will be resulted. The algorithms will therefore tend to select too few attributes for each cluster, which may be insufficient for clustering the records correctly. In some previous works on projected clustering (e.g. [1], [3] and [12]), the clusters are evaluated by using one or more of the following criteria:

1. How small are the distances between values at the selected attributes
2. How large is the number of selected attributes in the cluster
3. How large is the number of member records in the cluster

Intuitively, a small average distance between attribute values in a cluster indicates that the member records agree on a small range of values, which can make the records easily

*Research of the second author was supported partially by grants from the Research Grants Council of Hong Kong.

identifiable. A large number of selected attributes indicates that the records are similar at a high dimensional subspace, so they are very likely to belong to the same real cluster[1]. Finally, a large number of records in the cluster indicates there is a high support for the selected attributes, and it is unlikely that the small distances are merely by chance. All three are indicators for a good cluster, but there is actually a tradeoff between them. For example, given a fixed set of records, if we select only attributes that tend to make the average distance among records small, fewer attributes will be selected. Similarly, for a fixed distance requirement, putting more records into a cluster will probably decrease the number of attributes being selected.

To simplify the problem, some previous projected clustering algorithms try to optimize only some of the quantities, and leave the others as user parameters. CLIQUE [3] fixes the minimum density of each dense unit (related to criterion 1) by a user parameter, and searches for clusters that maximize the number of selected attributes (criterion 2).

PROCLUS [1] requires a user parameter l to determine the number of attributes to be selected (criterion 2). The algorithm is based on k-medoid, with additional logic for selecting on average l smallest-average-distance attributes for each cluster (criterion 1). With the aim of forming a specified number of disjoint clusters, the number of member records in each cluster (criterion 3) is irrelevant to the evaluation function.

ORCLUS [2] modifies the PROCLUS algorithm by adding a merging process of clusters and asking each cluster to select principal components instead of attributes. Essentially, the algorithm is still based on fixing the value of criterion 2 by a user parameter and minimizing the value of criterion 1 by algorithmically refining the partitioning of records.

The DOC and FastDOC algorithms [12] fix the maximum distance between attribute values (criterion 1) by a user parameter on the width of the bounding hypercubes, and search for clusters that maximize a function involving criteria 2 and 3.

In all these algorithms, there is a critical user parameter that determines the attributes to be selected, and hence the clusters to be formed. We will show in Section 4 that when wrong parameter values are supplied, the clustering accuracy can drop severely. Unfortunately, the correct values are rarely available in real applications. It would be preferable, therefore, to avoid introducing such critical parameters.

While we agree that it is hardly possible to have a projected clustering algorithm with no user parameter, an algorithm will be more usable if it has minimum dependency on the parameters. We elaborate the idea as the following three usability requirements:

1. The number and minimum quality of selected attributes are not directly specified by user parameters.

2. The clustering accuracy is not sensitive to small changes of parameter values.

3. Changing some data parameters (such as increasing the dimensionality of the dataset) does not increase the difficulty of finding proper user parameter values.

[1]We will use the term "real cluster" to mean the correct clusters defined according to domain knowledge.

In the remaining of this paper, we will present a highly usable hierarchical projected clustering algorithm[2] based on these three requirements. Although hierarchical clustering algorithms have a high time complexity intrinsically, its merging process is the key in avoiding the use of critical user parameters. We will justify the use of the hierarchical approach in more details later.

Before describing the algorithm, we will first introduce an index for measuring the relevance of an attribute to a cluster in Section 2. The index will be used in attribute selection of the algorithm as well as the construction of a similarity function. We will also describe the mutual disagreement problem and its prevention. In Section 3, the whole algorithm will be presented. Various kinds of experimental results will be presented in Section 4. A discussion on the performance issues will be given in Section 5, followed by a description of possible future works and the conclusions of the paper in Sections 6 and 7.

2. RELEVANCE INDEX

All the algorithms described so far calculate the relevance of an attribute to a cluster according to the distance between attribute values. It can thus be quantified by the variance of attribute values in the cluster, which we will call the "local variance" of the attribute in the cluster, as opposed to the "global variance", which is the variance of attribute values in the whole dataset. We find that low local variance does not necessarily imply high relevance. Consider the attributes a_1 and a_2 in Table 1. In cluster C, the local variance of a_1 is smaller than a_2. However, the global variance of a_1 is even smaller than the local variance. This means the values of a_1 in C are unlikely to be clearly distinguishable from other values, and therefore a_1 has a low relevance to C. On the other hand, although a_2 has a higher local variance in C, it has a large variance improvement from the whole dataset and thus a high relevance to C.

Variance of attribute values	a_1	a_2
Cluster C	0.2	0.5
Whole dataset	0.1	5.0

Table 1: Local variance does not necessarily imply relevance.

Based on the observation, we define the relevance index according to the global-to-local variance improvement. The relevance index of attribute a with respect to cluster C is defined as follows:

$$R(C, a) = 1 - \sigma^2(C, a)/\sigma^2(D, a),$$

where D is the whole dataset and $\sigma^2(S, a)$ is the variance of attribute a in the record set S.

For simplicity, we define $\sigma^2(C, a) = 0$ when there is only one record in C. We also assume no attribute has a global variance of 0 (otherwise, the attribute is useless and should be removed). The index has a maximum value of 1, and a higher index value indicates a higher relevance. When the local variance is just as high as the global variance, the attribute is not likely to help identify the records of the cluster, and the index value will be 0. This defines the baseline

[2]In this paper, whenever we use the term "hierarchical clustering", we always mean "agglomerative hierarchical clustering".

for an attribute to be regarded as relevant to a cluster. We will make use of this idea in the attribute selection process described later. The relevance index does not assume any data distribution model, so it is not restricted to any specific kinds of data.

When analyzing gene expression profiles, it is common to perform some preprocessing before applying the algorithms. It can be observed that the effect of relevance index is similar to performing standardization to each attribute. When relevance index is used, the concept of attribute relevance is automatically incorporated in the clustering process. This is important because according to our experience, standardizing the attributes can greatly improve the accuracy of projected clustering algorithms.

The relevance index will be used in two different parts of our algorithm: attribute selection and similarity calculation. In attribute selection, each cluster selects all attributes that have a relevance index above an internal threshold. The threshold is not supplied by user, but is determined directly from data and dynamically adjusted throughout the clustering process. The details will be given in the next section.

In hierarchical clustering, the merge order is determined by the similarity between different clusters. Traditional similarity measures may not work in projected clustering because two clusters can have different selected attributes. It is not well defined which attributes should be involved in the calculations. As in [2], we propose a function that makes use of the selected attributes of the merged cluster in determining the similarity between two clusters. Suppose C_1 and C_2 are two clusters, and if they merge to form C_n, A_{C_n} will be the new set of selected attributes. The similarity between C_1 and C_2 is then defined as

$$RSim(C_1, C_2) = \sum_{a \in A_{C_n}} R(C_n, a),$$

which measures the quality of C_n using criteria 1 and 2 (relevance and number of selected attributes) described in the last section.

2.1 Mutual disagreement

Using $RSim$, it is guaranteed that every next merge produces the cluster of highest quality among those producible by all the possible merges. However, we notice that some merges can be unfavorable due to a phenomenon that we call "mutual disagreement". This means two merging clusters are not really similar, but they receive a high similarity score. There are two situations that mutual disagreement will occur: unbalanced cluster size and unbalanced number of selected attributes. When a large cluster merges with a small cluster, the selection of attributes in the resulting cluster is mainly decided by the records of the large cluster. The local variance of values at the attributes can be very small in the new cluster even the records in the small cluster take completely different values from those in the large cluster. Similarly, if one cluster has many selected attributes and the other has few, the former may "lose" a substantial amount of its selected attributes if the clusters merge. The resulting cluster can have a "high quality" if the new cluster has small local variances at its selected attributes, but the merge is obviously unfavorable.

In order to oppose the effect of mutual disagreement, we construct a function to measure the mutual disagreement between two clusters, and disallow them to merge if the value is high. First, we define the relative relevance of attribute a to cluster C with respect to the potential new cluster C_n as follows:

$$R(C, a|C_n) = 1 - \frac{(\mu(C, a) - \mu(C_n, a))^2 + \sigma^2(C, a)}{\sigma^2(D, a)},$$

where $\mu(C, a)$ is the mean value of attribute a in C. The above formula calculates how relevant is attribute a to the records of cluster C in the new cluster C_n. If C is merged with a cluster with the same mean value and local variance at a, the two clusters agree on the values at a and the relative relevance will be high. On the other hand, if there is a large difference between their mean values, the relative relevance will be low even the two clusters both have a small local variance at a.

When mutual disagreement occurs, only one of the two clusters prefer the new mean value. The severity of mutual disagreement can thus be computed by the ratio of relative relevance values between the clusters:

$$MD(C_i, C_j) = \frac{1}{|A_{C_i} \cup A_{C_j}|} \sum_{a \in A_{C_i} \cup A_{C_j}} (1 - M(C_i, C_j, a)),$$

where A_{C_i} and $|A_{C_i}|$ are the set and number of selected attributes of C_i respectively, and

$$M(C_i, C_j, a) = \begin{cases} 0 & \text{if } R(C_i, a|C_n) \le 0 \text{ or } R(C_j, a|C_n) \le 0 \\ \frac{\min\{R(C_i,a|C_n), R(C_j,a|C_n)\}}{\max\{R(C_i,a|C_n), R(C_j,a|C_n)\}} & \text{otherwise} \end{cases}$$

MD takes a value between 0 and 1. A value of 0 means there is no mutual disagreement and a value of 1 means there is severe mutual disagreement. It can be easily seen that the function captures both situations described earlier. We will use MD to reject merges with heavy mutual disagreement. Suppose clusters C_i and C_j merge to form C_n with average relevance R_n at the selected attributes, the MD-adjusted average relevance will be $R_n \times (1 - MD(C_i, C_j))$. If the adjusted average relevance is lower than the threshold mentioned before, the merge will be rejected. The details will be given immediately in the next section.

3. THE ALGORITHM

In this section we describe our projected clustering algorithm HARP (a Hierarchical approach with Automatic Relevant attribute selection for Projected clustering). As indicated by the name, it does not depend on user parameters in determining the relevant attributes of each cluster. Instead, it tries to maximize the relevance index of each selected attribute and the number of selected attributes of each cluster at the same time. As discussed earlier, while both quantities indicate the quality of a projected cluster, maximizing one can have the other compromised. It is the merging process of hierarchical clustering that allows us to implement a dynamic threshold adjustment scheme that maximizes both criteria simultaneously.

At any time, there are two thresholds $|A|_{min}$ and R_{min} that restrict the minimum number of selected attributes for each

cluster and the minimum relevance index values of them. An attribute is selected by a cluster if and only if its relevance index with respect to the cluster is not less than R_{min}. Under this scheme, if an attribute is not selected by either of two clusters, it will also not be selected by the new cluster formed by merging them. However, if an attribute is selected by one or both of two clusters, it may or may not be selected in the new cluster, depending on the variance of the mixed set of values at the attribute. Two clusters are allowed to merge if and only if the resulting cluster has at least $|A|_{min}$ selected attributes. Initially, both thresholds are set at the highest possible values (the dataset dimensionality and 1 respectively), so that all allowed merges are very likely to involve records from the same real cluster. At some point, there will be no more possible merges with the current threshold values. This signals the algorithm to loosen the values and start a new round of merging. The process repeats until no more merging is possible, or a target number of clusters is reached. By dynamically adjusting the threshold values in response to the merging process, the number and relevance of the selected attributes are both maximized. Table 2 shows the details of the algorithm. Initially, each record is a cluster by itself. The whole clustering process is divided into threshold loosening steps. At the beginning of each step, $|A|_{min}$ and R_{min} are loosened linearly towards the baseline values (1 and 0 respectively). The merge score between each pair of clusters is calculated by a similarity function Sim, which can be the $RSim$ function defined earlier, or any other appropriate functions. All potential merges that have a high mutual disagreement or produce a cluster with fewer than $|A|_{min}$ selected attributes are ignored (Table 4). To speed up the clustering process, the merge scores are cached in score heaps similar to those used in [9]. Each cluster keeps a local heap that stores the merge scores between the cluster and all other clusters, and the best score is propagated to the global heap. After calculating the merge scores of all allowed merges, the information of the best merge will be extracted from the global heap and the two involved clusters will be merged to form a new cluster. The score heap entries involving the two clusters will be removed, and the merge scores between the new cluster and all other clusters will be inserted into the heaps. The process repeats until no more possible merges exist, and a new clustering step will begin.

Checking the usability requirements listed in Section 1, as the dynamic threshold adjustment mechanism involves no user parameter, HARP readily satisfies all of them. This is an advantage of the hierarchical approach. The algorithm only takes the target number of clusters k as input, which is an acceptable parameter and is required by many clustering algorithms. In addition, the whole clustering process can be logged as a dendrogram. This offers an option for users to set k to 1 to produce a single merge tree of all records, and get the results at different k values by cutting the merge tree correspondingly. This prevents running the algorithm many times, which is unavoidable for most non-hierarchical algorithms. The dendrogram also shows the relative similarity between different records, which can be very useful in analyzing gene expression profiles with unclear cluster boundaries [7].

These benefits justify the use of the hierarchical approach in spite of its high time complexity. This is especially true when accuracy is the first priority and the dataset size is not

```
// d: dataset dimensionality
// |A|_min: min. no. of selected attributes per cluster
// R_min: min. relevance index of a selected attribute
Algorithm HARP(k: target no. of clusters)
Begin
1    // Initially, each record is a cluster
2    For step := 0 to d − 1 do {
3       |A|_min := d − step
4       R_min := 1 − step/(d − 1)
5       Foreach cluster C
6          SelectAttrs(C,R_min)
7       BuildScoreHeaps(|A|_min,R_min)
8       While global score heap is not empty {
9          // C_b1 and C_b2 are the clusters involved in the
10         // best merge, which forms the new cluster C_n
11         C_n := C_b1 ∪ C_b2
12         SelectAttrs(C_n,R_min)
13         Update score heaps
14         If clusters remained = k
15            Goto 18
16      }
17   }
18   Output result
End
```

Table 2: The HARP algorithm.

```
Procedure SelectAttrs(C,R_min)
Begin
1    Foreach attribute a
2       Select a if R(C, a) ≥ R_min
End
```

Table 3: The attribute selection procedure.

```
Procedure BuildScoreHeaps(|A|_min,R_min)
Begin
1    Foreach cluster C_i do {
2       Foreach cluster C_j ≠ C_i do {
3          C_n := C_i ∪ C_j
4          SelectAttrs(C_n,R_min)
5          // |A_Cn| is the no. of selected attributes of C_n
6          If |A_Cn| ≥ |A|_min and
           (Σ_{a∈A_Cn} R(C_n,a) / |A_Cn|) × MD(C_i, C_j) ≥ R_min
7             Insert Sim(C_i, C_j) into local heaps of C_i and C_j
8       }
9       Insert max_j Sim(C_i, C_j) into the global heap
10   }
End
```

Table 4: The score heaps building procedure.

very large. In order to cope with large datasets, we will also discuss some methods to increase the speed performance of HARP later.

Finally, we briefly describe the outlier handling mechanism of HARP. We implement a scheme similar to that in [8] with two phases. Phase one is performed when the number of clusters remained reaches a fraction of the original number. Clusters with very few records are removed. Phase two is performed near the end of clustering. All clusters that are much smaller than others are removed. As pointed out in [8], the time to perform phase one outlier removal is critical. Having it too early may remove many non-outlier records, while having it too late may have some outliers already merged into clusters. We add a record fill back step to reduce the error caused by an incorrect time to perform phase one outlier removal. Phase one is performed relatively earlier so that most outliers are removed, possibly with some other records also removed. When phase two is performed, the removed records try to fill back to the most similar cluster subject to the current threshold requirements. Due to the requirements, outliers are unlikely to be filled back. In our experiments, the fill back step usually increases the accuracy of outlier removal.

4. EXPERIMENTS

In this section we present various experimental results. We will start with a brief description of the datasets, algorithms, similarity functions and performance metrics used in the experiments.

4.1 Datasets, algorithms, similarity functions and performance metrics

Synthetic datasets: The parameters used generating synthetic datasets are summarized in Table 5.

Data parameter	Values used
Number of records (N)	500-10000
Dataset dimensionality (d)	20-100
Number of clusters (k)	5
Cluster dimensionality	20-60% of d
Cluster size	15-25% of N
Local variance of relevant attributes	0-1% of the domain
Artificial data errors (e)	10-20%
Artificial outliers (o)	0-5%

Table 5: Data parameters of the synthetic datasets.

When generating a dataset, the size of each cluster and the domain of each attribute were first determined randomly. Each cluster then randomly picked its relevant attributes, where the number of relevant attributes was chosen from the allowed range randomly. For each relevant attribute of a cluster, a local mean was chosen randomly from the domain. Each record in the cluster determined whether to follow the relevant attribute values according to the data error rate. This was to simulate experimental and measurement errors. If it was determined to follow, an attribute value would be chosen from a Gaussian distribution with the local mean and a variance chosen from the variance parameter. Otherwise, values were generated randomly as in irrelevant attributes.

We used the synthetic datasets to compare the clustering results of HARP and different projected clustering algorithms. We chose PROCLUS [1], ORCLUS [2] and FastDOC [12] for the study, because they have reasonable worst case time and can produce disjoint clusters. FastDOC creates clusters one at a time. We used it to produce disjoint clusters by removing the clustered records before forming a new cluster. After forming the target number of clusters, the unclustered records were treated as outliers. To be fair to each algorithm, we performed standardization on each attribute of the datasets[3]. We like to note that the sizes of the synthetic datasets imitate the size of normal gene expression datasets.

Real dataset: Except the performance on synthetic data, we also want to study whether projected clustering is applicable to real gene expression profiles. In this paper we present the results on one of the datasets used in studying distinct types of diffuse large B-cell lymphoma (Figure 1 of [4]). The dataset contains 96 samples, each with 4026 expression values. We clustered the samples with the expression values of the genes as attributes. The samples are categorized into 9 classes according to the category of mRNA sample studied. We will use the class labels to evaluate the clustering results.

For this dataset, we used PROCLUS and HARP as the representatives of projected clustering algorithms as they need less execution time. We compared the results with non-projected clustering algorithms, including a hierarchical algorithm (HARP with attribute selection and mutual disagreement prevention disabled), two partitional algorithms CLARANS [11] and KPrototype [10], and CAST [5], a popular algorithm for clustering gene expression profiles. We believe our choice of algorithms covers a wide variety of clustering approaches. The experiments were performed on both raw and attribute-standardized data.

Similarity functions: We used Euclidean distance and Pearson correlation as the similarity functions for non-projected algorithms. Pearson correlation is commonly used in clustering gene expression datasets (see for example [7]). CAST requires a similarity function with known value range, so only Pearson correlation was used. As all projected clustering algorithms define attribute relevance according to value distance, Pearson correlation was not used. For the two hierarchical algorithms, the $RSim$ function was also used.

Performance metrics: We will use the Adjusted Rand Index [14] as the performance metric for clustering accuracy. It is based on the Rand Index [13], with the expected baseline value also taken into account. Denote U as the partition of records according to the real clusters based on domain knowledge and V as the partition of records in a clustering result. Let a be the number of record pairs that are in the same cluster in both U and V, b be the number of pairs that belong to the same real cluster in U, but are wrongly put into different clusters in V, c be the number of pairs that belong to different real clusters in U, but are wrongly put into the same cluster in V, and d be the number of pairs that are in different clusters in both U and V. The Adjusted Rand Index is then defined as follows:

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

The Adjusted Rand Index gives a value of 1 when two partitions are identical (in our case, one partition is the known

[3]The mean and standard deviation of the values of each attribute become 0 and 1 respectively.

clustering and the other is the clustering result), and 0 when the Rand Index equals the expected value of a random partition. Between the two extremes, a higher value indicates a higher similarity between the partitions. When evaluating the selected attributes of a projected clustering algorithm, the precision and recall measures commonly used in machine learning will be used. For each cluster, precision measures the number of selected relevant attributes divided by the total number of selected attributes. Recall measures the number of selected relevant attributes divided by the total number of relevant attributes. The reported values of a clustering result are the average of all the clusters.

Practical setup: All algorithms that make use of random numbers (i.e., all except HARP and non-projected hierarchical) were run 5 times on each dataset. The results that give the best values of algorithm-specific criterion functions were reported.

4.2 Comparing the projected clustering algorithms

We first show the clustering results on the synthetic datasets. We started with a small dataset with 5% outliers. It was noticed that PROCLUS, ORCLUS and FastDOC all discarded too many records as outliers and made the comparison difficult. Therefore in the subsequent datasets, we did not introduce any artificial outliers. Due to the way that FastDOC produces disjoint clusters, its outlier removal function could not be disabled. The results are summarized in Table 6. In the table, the numbers outside the brackets are the Adjusted Rand Indices and the bracketed values are the percentage of records discarded as outliers. Except HARP, the best and average results of each algorithm with different parameter values are shown. We used 20 sets of parameters for FastDOC per dataset, and 10 sets for PROCLUS and ORCLUS each.

In all datasets, the results of HARP are always among the best ones. More importantly, as reflected by the low accuracies of the average results, the other three algorithms only achieved their best results when correct parameter values were used. For example, PROCLUS only attained its best result when l was very close to the correct number. Also, the accuracies of all three algorithms are very sensitive to the parameter inputs. A small perturbation brought down the accuracies a lot. We notice that it is especially difficult to set the parameters of FastDOC as each relevant attribute can have a different local variance value. The projected clusters do not appear to take the form of hypercubes. It also has a tendency of discarding too many records as outliers. In comparison, HARP produced each result in a single run without the need to try many different parameter values.

Another important performance metric of the projected clustering algorithms is the accuracy of the selected attributes. Table 7 compares the selected attributes of HARP and the best results of PROCLUS. Since FastDOC discards too many records and ORCLUS selects principal components instead of attributes, it is difficult to compare their selected attributes with those of HARP.

The table shows that HARP has excellent recall values, indicating its ability to select the relevant attributes. The precision is relatively lower since HARP begins with selecting all attributes and stops once the target number of clusters is reached. Some extra attributes may be selected, but they have only limited impact on the clustering accuracy. In contrast, PROCLUS has lower recall values, showing that it often misses some relevant attributes. The clustering accuracy was also affected. In other results of PROCLUS in which the l values used deviate from the correct one, the attribute selections were even more inaccurate.

4.3 Analyzing real data

For the lymphoma dataset, 56 results were produced by the 6 algorithms. Table 8 shows the best results of each algorithm-similarity pair sorted by Adjusted Rand Index.

Adjusted Rand Index	Algorithm	Similarity
0.70	HARP	*RSim*
0.64	PROCLUS	Euclidean
0.59	HARP	Euclidean
0.55	CLARANS	Pearson
0.55	CAST	Pearson
0.48	KPrototype	Euclidean
0.44	Hierarchical	Pearson
0.37	KPrototype	Pearson
0.37	Hierarchical	*RSim*
0.05	Hierarchical	Euclidean
0.05	CLARANS	Euclidean

Table 8: The best results of each algorithm-similarity pair on the lymphoma dataset.

HARP and PROCLUS have the best performance, which suggest that projected clusters may exist in the dataset. However, among the 22 results of PROCLUS produced by using different l values, the average Adjusted Rand Index is only 0.27 and only 2 results have an index value larger than 0.5. This confirms that PROCLUS requires accurate parameter values to produce satisfactory results. For HARP, no parameter is needed to guide its attribute selection.

The result of HARP with *RSim* has on average 3022 selected attributes per cluster, accounting for 75% of all attributes. We sorted the selected attributes of each cluster according to their relevance indices, and search for the ranks of some signature genes that are known to be meaningful to the clusters (Figure 2 of [4]). It was found that many signature genes got a high rank in the list. For example, it is known that DLBCLs in general had higher expression of the genes in the proliferation signature [4]. The clustering result of HARP contains a large cluster mainly consists of DLBCLs with 3715 selected genes. The genes in the proliferation signature received high ranks in the list. For example, Ki67, p16 (clone 550316) and SOCS-1 (clone 430097) received ranks 7, 9 and 12, with relevance indices 0.86, 0.85 and 0.84 respectively.

The advantage of *RSim* over Euclidean distance is also observed from the results of HARP and the non-projected hierarchical algorithm.

We notice that similar to FastDOC, CAST has many records left unclustered with most parameter values. The result reported in Table 8 is the best one with less than 10% unclustered records. In general the results produced by CAST are good, but the large number of unclustered objects can be a potential problem.

5. PERFORMANCE ISSUES

The worst case time complexity of HARP can be shown to be $O(N^2 d(d + logN))$, where N and d are the dataset size

Algorithm	$N = 500$, $d = 20, o = 5\%$	$N = 500$, $d = 20, o = 0\%$	$N = 10000$, $d = 20, o = 0\%$	$N = 500$, $d = 100, o = 0\%$
FastDOC (best)	0.94 (35.6%)	0.95 (75.2%)	0.56 (72.90%)	0.91 (82.80%)
FastDOC (avg.)	0.60 (59.1%)	0.46 (62.5%)	0.38 (63.00%)	0.43 (91.30%)
HARP	0.97 (5.4%)	0.84 (0.0%)	0.98 (0.00%)	0.99 (0.00%)
ORCLUS (best)	1.00 (99.0%)	0.86 (0.0%)	0.86 (0.00%)	1.00 (0.00%)
ORCLUS (avg.)	0.68 (74.9%)	0.41 (0.0%)	0.65 (0.00%)	0.19 (0.00%)
PROCLUS (best)	0.86 (50.8%)	0.76 (0.0%)	0.82 (0.00%)	0.93 (0.00%)
PROCLUS (avg.)	0.63 (34.0%)	0.55 (0.0%)	0.63 (0.00%)	0.75 (0.00%)

Table 6: Clustering results on synthetic datasets.

Dataset	Avg. Cluster Dimensionality	HARP			PROCLUS (best)		
		Avg. Sel. Attr.	Precision	Recall	l	Precision	Recall
$N = 500, d = 20, o = 5\%$	11	13.2	83.3%	100%	10	72.1%	61.9%
$N = 500, d = 20, o = 0\%$	8.8	11.6	75.1%	100%	10	77.6%	90.9%
$N = 10000, d = 20, o = 0\%$	8.4	9.2	90.8%	100%	10	85.6%	100%
$N = 500, d = 100, o = 0\%$	28.8	44	65.4%	100%	30	83.2%	85.9%

Table 7: Accuracy of the selected attributes.

and dimensionality respectively. The execution time with various N and d values are shown in Figures 1 and 2, which show that the theoretic time complexity is a loose upper bound of the real time complexity. For normal gene expression profiles, HARP is always faster than ORCLUS and usually faster than FastDOC for some parameter values. We also want to emphasize that in order to obtain a satisfactory result, non-deterministic algorithms have to be run on each dataset a number of times. Since HARP gives deterministic results, repeated runs are not necessary and the actual execution time is comparable to many other algorithms when running on normal gene expression datasets.

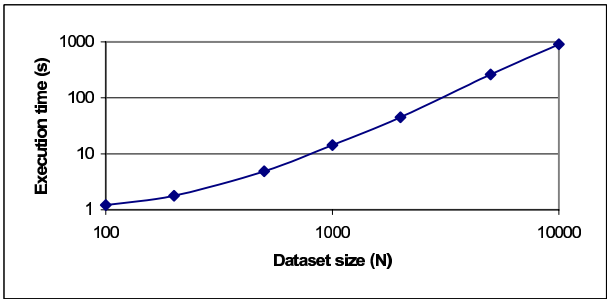

Figure 1: Execution time of HARP with various N.

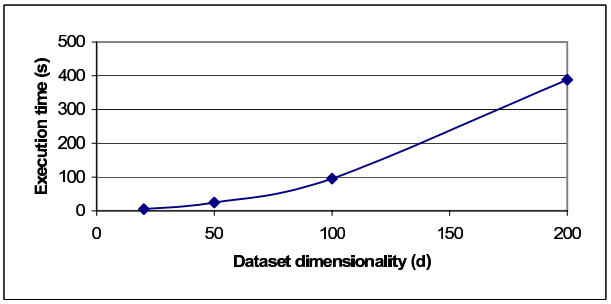

Figure 2: Execution time of HARP with various d.

When N is very large, HARP can be run on a random sample of records. After completion, all records in the dataset are assigned to the most similar cluster. When d is very large, HARP can be modified to use a smaller number of threshold loosening steps to reach the baseline values. To study the feasibility of these approaches, we performed some experiments on the two synthetic datasets with 10000 records and 100 attributes. The relative execution time and accuracy are shown in Figures 3 and 4.

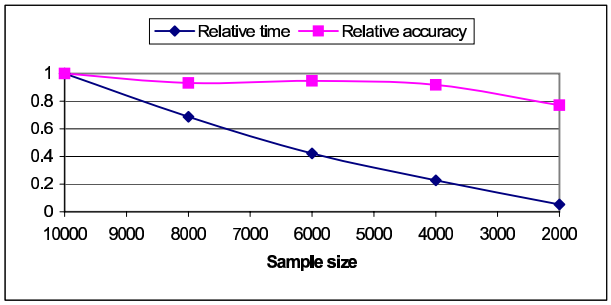

Figure 3: Relative execution time and accuracy of HARP with various sample sizes.

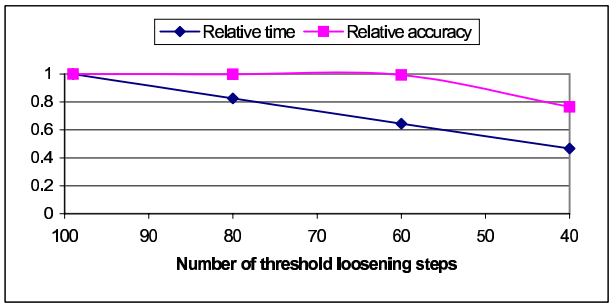

Figure 4: Relative execution time and accuracy of HARP with various no. of threshold loosening steps.

A significant speed up is obtained with a reasonable loss of

accuracy when the sampling and reduction of step count are moderate. The approaches are shown to be feasible.

6. FUTURE WORKS

The dynamic threshold adjustment scheme of HARP uses a simple linear loosening. While it works pretty well according to the experimental results, we believe a more advanced scheme is possible, which may improve the speed performance and further boost the accuracy. Another interesting topic to study is an attribute selection procedure that is not based on value distance. For example, the relevance of an attribute to a cluster can be measured by its correlation with other attributes of the cluster. This can be useful in detecting genes that are co-regulated, but with different magnitudes or even directions. Some studies have been started in the community, and we are looking for a more general framework.

7. CONCLUSIONS

In this paper, we have described a projected clustering algorithm HARP with high usability. Unlike some existing projected clustering algorithms, it does not depend on user inputs in determining the relevant attributes of clusters. This is very important in real applications, since the correct values for the parameters are usually unknown and when wrong values are used, the performance can drop significantly. HARP makes use of relevance index and the merging process of hierarchical clustering to dynamically adjust the internal thresholds and decide the attributes to be selected for each cluster. According to the experimental results, HARP performs much better than the projected clustering algorithms being compared in terms of parameter-sensitivity, and is in general superior in terms of clustering accuracy and the selection of relevant attributes. Experiments on the lymphoma dataset also suggest that projected clustering can be useful in analyzing gene expression profiles.

8. ACKNOWLEDGEMENTS

We would like to thank Kei-Hoi Cheung, Antoine Danchin, Raymond Ng, Walter Larry Ruzzo and three anonymous reviewers for their precious comments.

9. REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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*, 403(6769):503–511, 2000.

[5] A. Ben-Dor and Z. Yakhini. Clustering gene expression patterns. In *Proceedings of the third annual international conference on Computational molecular biology*, pages 33–42, 1999.

[6] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science*, 1540:217–235, 1999.

[7] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868, 1998.

[8] 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.

[9] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *15th International Conference on Data Engineering*, 1999.

[10] 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.

[11] 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.

[12] 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.

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

[14] K. Yeung and W. Ruzzo. An empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.