Abstract of thesis entitled

“HARP: A Practical Projected Clustering Algorithm for Mining Gene Expression Data”

Submitted by

Kevin Yuk-Lap YIP
(葉旭立)

for the degree of Master of Philosophy
at The University of Hong Kong
in December 2003

In high-dimensional data, the similarity between different objects of a cluster may only be reflected in a certain subspace. In microarray gene expression data, this phenomenon could occur when a set of co-regulated genes have similar expression patterns only in a subset of the testing samples in which certain regulating factors are present. As a result, the expression patterns of the genes could appear to be dissimilar in the full input space. Traditional clustering algorithms that utilize such similarity values in determining object similarity might therefore fail to identify the clusters.

In recent years a number of algorithms have been proposed to identify this kind of projected clusters. Many of them require the input of some parameter values that are hard for users to supply, and clustering accuracy can be seriously affected if incorrect values are used. In gene expression data analysis it is rarely possible to obtain precise estimations of the parameter values, and this causes practical difficulties in applying the algorithms to real data.

This study provides a thorough analysis of the proposed projected clustering algorithms and suggests some reasons for their heavy parameter dependency. Based on the analysis, a new algorithm is proposed to exploit the clustering status in adjusting the internal thresholds dynamically without the assistance of user parameters. This allows automatic processing of large amounts of data without user intervention. The algorithm is also extended to handle pattern-based clustering and the production of non-disjoint clusters, which are useful when analyzing gene expression datasets that involve samples taken at different time points.

The results of extensive experiments on both synthetic and real data show that the new algorithm outperforms some traditional and projected clustering algorithms in terms of both accuracy and applicability. It is also capable of identifying clusters that make both statistical and biological sense.

HARP: A Practical Projected Clustering Algorithm for Mining Gene Expression Data

by

Kevin Yuk-Lap Yip
(葉旭立)

A thesis submitted in partial fulfillment of the requirements for
the degree of Master of Philosophy
at The University of Hong Kong.

December 2003

Declaration

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

. . . .

Kevin Yuk-Lap Yip

December 2003

Acknowledgements

It would have been impossible for me to complete the two years of study if I had not received the various helps and supports from the many benefactors. I would like to express my deepest gratitude to all of them.

Thank God for giving me this opportunity to experience another style of living and for guiding and protecting me during the two years. Thank my family members for their full support of my decision to study again after working for a few years.

Thank my supervisor Dr. David Cheung for his guidance in all aspects during my study and for providing me a lot of exposures to the research community. I also thank Dr. Michael Ng for teaching me so much through the many lengthy discussions.

I would like to thank for the financial support from the Hong Kong and China Gas Company Limited Postgraduate Scholarship.

I have enjoyed very much the various research and recreational activities at the HKU-Pasteur Institute. I would like to thank all its members, especially my supervisor Prof. Antoine Danchin, who is willing to teach me from the very fundamental level.

I also had a great time in YCMI. The members of YCMI are friendly and helpful, and our collaboration is solid. I would like to give special acknowledgement to Dr. Kei Cheung, who took care of my whole journey, and have been giving me continuous advise and helps.

Thank Prof. Raymond Ng, Prof. Larry Ruzzo, Dr. Zhang Xue Wu and the anonymous reviewers of my research articles for their invaluable comments on my works.

I have gained a lot from the database research group. Thank Dr. Ben Kao, Dr. Nikos Mamoulis and Dr. C. L. Yip for all the teachings and helps. I also thank the student members and friends of DB group, especially Jolly Cheng, Felix Cheung, Sherman Chow, Ho Wai Shing, Eric Lo and Ivy Tong for the works and fun that we have created together.

Thank all the faculty, staff and students of the CSIS department who have enriched my research life. I would like to give special thank to Dr. Yiu Siu Ming for his advice during my early stage of study and his concern for me. Thank the staff members of CECID, especially Albert Kwan, who have been working with me on the web service project. I would also like to thank my roommates, in particular Cheng Lok Lam, Chow Yuk and Vivian Kwan, who have accompanied me during the numerous weekdays and weekends.

Finally, I must say thank you to all Katso friends, especially the members of the 45th executive committee and the Engineering cell. They have given me supports in many different ways.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The main theme of this thesis is to study the feasibility of extracting useful information from gene expression profiles by a relatively new data mining approach known as projected clustering. This is a multi-disciplinary topic, involving research efforts from areas such as data mining, applied mathematics, genetics and genomics. A complete introduction to the topic may require a few dictionary-sized books, which obviously cannot fit into this thesis. Rather, this introductory chapter is being kept as concise as possible to cover just enough material for the understanding of this text. Readers interested in a deeper introduction of the topic are advised to read the references from the corresponding areas.

This chapter consists of two main parts. The first part provides a short overview of the objectives and methods of data mining, and quickly moves to the topic of clustering and finally projected clustering. The second part starts with an introduction to bioinformatics in general, and then narrows down to microarray technology and gene expression profiles, and finally focuses on some clustering methods proposed for gene expression profile analysis. The last section of this chapter describes the outline, main contributions and scope of this thesis.

1.1 Data Mining

An era has come when the rate of data generation is much higher than the maximum data analyzing speed of human experts. There is a great need to extract a succinct set of interesting patterns from the sea of data systematically and automatically so that people can benefit from it. This process is known as data mining. Just like mining precious gold or silver from inexpensive rocks and sand, data mining "digs out" valuable information from the numerous otherwise useless data. It is a complex multi-step knowledge discovery process that involves 1) data cleaning, 2) data integration, 3) data selection, 4) data transformation, 5) data mining, 6) pattern evaluation and 7) knowledge presentation [36]. The narrowed meaning of data mining in step five is the application of certain mining algorithms to extract information from preprocessed data. This is the main focus of this thesis, although some issues related to the other steps will also be discussed. In the remaining of this text, the narrowed meaning of data mining is assumed.

Data mining has some important characteristics that derive a number of requirements for the mining algorithms:

- The amount of data to be mined is huge, so all practical mining algorithms should be efficient and scalable.
- As new data is evolving from time to time, the mining process should be largely automated and involve minimum user intervention. It is necessary and absolutely reasonable to ask users to evaluate the interestingness of the mined patterns, but requiring frequent human feedback during the mining process is generally unacceptable.
- Human users can only interpret results of reasonable sizes, so mining algorithms should intelligently select the most interesting results to report.
- To further assist users to interpret the results, graphical visualization tech-

niques may be used to display them in a more manageable and intuitive fashion.

Each mining algorithm assumes some types of patterns to be mined from data, but the actual patterns are not known before the mining process. For example, in association rule mining [6], the association rules to be mined are patterns in the form $A \Rightarrow B$, which means there is a high support and confidence that when the items in itemset A occur, the items in itemset B also occur. The actual association rules (i.e., the items in A and B of each rule) that exist in a dataset are, however, not known to users.

There are a few popular data mining approaches, including association rules mining [6], classification [53], clustering [45], emerging pattern mining [24], and sequence mining [7]. The approaches are complementary to each other and have different applications as they work on different types of data, require different amount of domain knowledge, and produce different kinds of results. Clustering is the focus of this thesis. As to be explained later, it has some properties that make it suitable for gene expression data analysis.

1.2 Clustering

Clustering is a process to group similar objects together. Before directly jumping into the detailed discussion of clustering, it is instrumental to spend some time on the format of data that can be clustered. In the following definitions, the preferred terms of some concepts appear first and some alternative terms that have the same or similar meanings are listed in brackets. The preferred terms are used most frequently in this thesis, while the alternative terms are occasionally used when they can better illustrate some ideas.

Each *dataset* to be clustered is represented by a *set* (*table, matrix*) that consists of *objects* (*records, rows, tuples, instances*). Each object is described by its *projected values* (*projections, attribute values*) on the different *dimensions* (*attributes, columns, features*) of the dataset. Unless otherwise stated, all datasets are assumed to contain continuous numeric attributes with no missing values. This means all projected values are real numbers.

A *cluster* is defined as a non-empty subset of the objects, which are called the *members* of the cluster. The *centroid* of a cluster is a virtual object with its projected value on each dimension equal to the arithmetic mean of the projected values of all the cluster members on the dimension[1]. Two clusters are *disjoint* (*non-overlapping*) if they contain no common objects, and a set of clusters is disjoint if all clusters in it are pairwise disjoint. A clustering algorithm is said to produce disjoint clusters if the clusters that it produces are always disjoint. Otherwise, the algorithm is said to produce *non-disjoint* (*overlapping*) clusters, even the clusters are not always non-disjoint. Some objects that do not fit into any cluster can be left unclustered. They are called the *outliers* of the dataset, which are reported on a separate outlier list.

The goal of clustering is to partition the objects into clusters so that objects in the same cluster are similar to each other, but dissimilar to objects not in the cluster. The similarity between two objects is measured by a *similarity function*[2], which specifies the properties of the clusters to be formed. Some commonly used similarity functions include L_m (Minkowski) distance, cosine correlation and Pearson correlation. The similarity between two clusters can be computed as the similarity between the most similar (*single link*) or the most dissimilar (*complete link*) objects in the two clusters, the average of all inter-cluster object similarities (*average link*), or the similarity between the two centroids (*centroid link*). Depending on the particular application need, other object/cluster similarity functions can also be defined.

[1]In practice, some clustering algorithms do sometimes produce empty clusters. The centroid of an empty cluster is represented by the null object, which is handled by some special logic of the algorithms.

[2]In this thesis, we will not make a distinction between similarity functions and dissimilarity functions.

A huge number of clustering algorithms have been proposed, which probably exceeds the number of algorithms proposed for all other data mining approaches. The clustering algorithms can be briefly classified according to their ways of cluster identification:

- Hierarchical algorithms [45]: there are two main types, agglomerative and divisive. Agglomerative algorithms treat each data object as a singleton cluster. The algorithms repeatedly identify the two most similar clusters and merge them into a larger cluster until certain stopping criteria are reached. Conversely, divisive algorithms put all objects into a single cluster initially. Each time a cluster is divided into two smaller clusters so that dissimilar objects are separated to different clusters. In general agglomerative algorithms are more popular as divisive algorithms could have exponential time complexity [45]. In this thesis, when the term "hierarchical clustering algorithm" is used alone, it implicitly means "agglomerative hierarchical clustering algorithm".

- Partitional algorithms: these algorithms select some seeds as the representatives of the clusters. Every object in the dataset is assigned to the most similar seed to form clusters. The goodness of the clusters (and thus the seeds) is evaluated by an objective function. Different sets of seeds are tried, and the set that yields the best objective function value (objective score) is reported. There are two major types of partitional algorithms: k-means [38] and k-medoids [55]. They differ by the choice of seeds: k-means algorithms use centroids as seeds, while k-medoids algorithms select objects from the dataset as seeds.

- Density-based algorithms [28]: this kind of clustering algorithms regards a cluster as an arbitrarily shaped structure containing a high density of objects. The algorithms usually determine a small high-density region as the initial cluster, and then progressively expand the cluster by including objects in the neighboring dense regions into the cluster.

- Model-based algorithms [29]: these algorithms construct statistical models for clusters, and try to fit the data objects into the models to deduce the best parameter values to use.

There are also many other clustering algorithms that are variations or hybrids of the above classical algorithms. Besides the above generic categories, special clustering algorithms have also been invented to address many important issues, such as fuzzy clustering [45], limitation of memory [73], clusters of irregular shapes [33], datasets with categorical [34] or mixed types [41] of attributes, outliers and missing values handling [45] and visualization of clustering results [10]. Some of these issues will be discussed later.

For there are so many kinds of algorithms, there must be some ways to compare the effectiveness of different algorithms on certain applications. This can be done by theoretical reasoning and empirical studies. In the former way, the characteristics of each algorithm are matched against the data properties and specific requirements of the application. For example, in later parts of this thesis, theoretical reasoning will be used to show that projected clustering algorithms are in theory superior to some other kinds of clustering algorithms in detecting clusters in high dimensional data.

The other effectiveness indicator is the experimental results. Normally both synthetic and real datasets are used to test the algorithms. Synthetic datasets, generated according to the assumed data models, capture the most crucial data properties that the clustering algorithms have to be able to handle, including those properties that are less commonly found in real datasets. On the other hand, real datasets are used to detect any discrepancies of the data models from the actual data characteristics. They also test the stability, usability and efficiency of the algorithms in real situations.

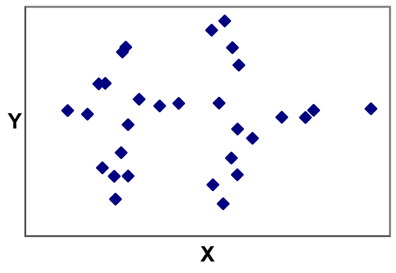
No matter synthetic or real datasets are used, there are two basic techniques to evaluate the clustering results (i.e., the performance of a clustering algorithm on a particular dataset). The two techniques are called internal validation and external validation. Internal validation concerns to what degree the clusters produced fit the assumed model. For example, in the k-means model [38], each cluster is a spherical structure with member objects close to the centroid. A natural evaluation function for internal validation is thus the average within-cluster distance to centroid. The better (smaller in this case) is the evaluation score, the more likely the model fits the data and the algorithm produces clusters according to the model.

In external validation, some domain knowledge about the dataset is utilized to evaluate the clustering results. For example, some datasets contain known "class labels" of data objects, such as the object types assigned by a domain expert. These labels suggest the members of the desired clusters, which can be used as a "gold standard" to evaluate the clusters formed by an algorithm. The more similar are the two sets of clusters (measured by some statistical functions), the more likely the algorithm works well in the particular application. It should be noted, however, that the class labels are only used in result evaluation but do not participate in the clustering process.

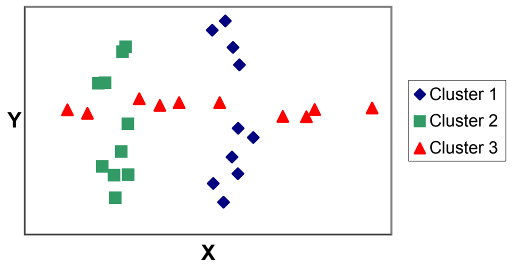
1.3 Projected Clustering

In recent years, a special branch of clustering problems called projected clustering has been receiving a lot of attention from various communities due to its ability to analyze high-dimensional datasets, which are very common in some areas. In projected clustering, clusters exist in subspaces of the input space defined by the dimensions 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 distinct subspace.

To illustrate the idea of projected clusters, consider the data points in Figure 1.1a. Although the distribution of points suggests some underlying struc-



(a) A set of 2D points.



(b) 1-D projected clusters.

Figure 1.1. An example illustrating the idea of projected clusters.

Stud.	Lang.	Bio.	Comp.	Music
1	89	50	82	88
2	90	70	66	91
3	91	83	50	90
4	70	90	92	76
5	55	65	90	63
6	79	54	99	50

(a) Data records.

Stud.	1	2	3	4	5
2	26				
3	46	21			
4	47	41	50		
5	46	51	63	32	
6	43	61	61	46	31

(b) Distance between different records.

Figure 1.2. Projected clusters in a virtual examination score dataset.

tures, it is hard to unambiguously partition the points into clusters. The hidden relationship between the points is revealed in Figure 1.1b, where the points of different clusters are given different shapes. By projecting the points onto appropriate subspaces (clusters 1 and 2 onto the one-dimensional space formed by X, cluster 3 onto the one-dimensional space formed by Y), the cluster structures become apparent.

A relational example is shown in Figure 1.2a, which contains the examination scores of six students in four subjects. Figure 1.2b shows the dissimilarity between each pair of students based on Euclidean distance. Again, it is not easy to observe the hidden clusters. By identifying the proper subspaces, students 1-3 are found to form a cluster that has special talent in language and music, while students 4-6 form another cluster that is good at computer.

Projected clusters can appear in various kinds of real data. The projected clustering approach has been successful in application areas including computer vision [59], food categorization [48], and gene expression data analysis (e.g. [18]). It has also been suggested that projected clustering has potential applications in e-commerce [68].

We now define projected clusters more formally[3]. Given a dataset D with N objects and a set V of d input dimensions, a *projected cluster* C_I contains N_I objects and is defined in a d_I-dimensional subspace formed by the set V_I of dimensions, where $V_I \subset V$. In the subspace, the members of C_I are similar to each other according to a similarity function, but dissimilar to other objects not in C_I.

Since a few new concepts are considered, we need to introduce some more terms for the sake of discussion. d_I is called the *dimensionality* of cluster C_I, which is the size of the set of *relevant dimensions* V_I of the cluster. In [4], the dimensions in V_I are along the directions of the principal components of C_I, which are not necessarily elements of V. As in most other studies (e.g. [3, 59]), we adopt a more restrictive definition that requires each V_I to be a subset of V as the clustering results are more understandable by human. Based on this definition of relevant dimensions, the set of *irrelevant dimensions* of cluster C_I is defined as the set $V - V_I$. A dimension can be relevant to zero, one, or more clusters.

A projected cluster is usually modeled as a combination of local distributions along the relevant dimensions and global distributions along the irrelevant dimensions. This means the projected values of the cluster members on the relevant dimensions form some patterns unique to the cluster, while the projected values on the irrelevant dimensions are indistinguishable from the values from other clusters to which the dimensions are also irrelevant. Certainly, the more irrelevant dimensions a cluster has, the less similar are its members in the full

[3]A list of symbols used in this thesis can be found in Appendix B.

input space. It has been shown in [15] that under a wide variety of conditions, the distance to the nearest object approaches the distance to the farthest object as the number of dimensions on which the values are generated from a global distribution increases. The effect can become quite severe when there are only 10-15 dimensions, much lower than the dimensionalities of the high dimensional datasets to be considered in this thesis. This implies that the similarities between different objects of the same cluster due to the relevant dimensions can be washed out by the irrelevant dimensions. In other words, the members of a cluster can hardly be discovered if the relevant dimensions are not identified.

Projected clusters that are defined according to some domain knowledge (e.g. known class labels) are called the *real clusters* of the dataset and the corresponding relevant dimensions the *real relevant dimensions* of the real clusters. The general term *clusters* will be used to mean the object groups identified by a clustering algorithm. We choose the simple term "cluster" instead of "projected cluster" due to the frequent occurrence of the concept in the text and the fact that a non-projected cluster is actually a special case of a projected cluster with all input dimensions being relevant to it. A cluster is *correct* if it contains objects all from the same real cluster, and *incorrect* otherwise.

For each cluster, a projected clustering algorithm determines its relevant dimensions by finding a subspace in which the cluster members are similar to each other, but dissimilar to other objects outside the cluster. For simplicity, clustering algorithms that have the ability to identify the relevant dimensions of each cluster are called generically the *projected algorithms*. All other clustering algorithms are termed the *non-projected algorithms*.

The dimensions regarded as relevant to a cluster by a projected algorithm are called the *selected dimensions* of the cluster. When object similarity is measured by a distance metric, a dimension is likely relevant to a cluster if the projections of the cluster members on the dimension concentrate at a specific small region containing few or no projections of other objects. The region is thus a *signature* of the cluster, which can be used to identify the cluster members. We call this kind of clustering that measures object similarities based on a distance function a *distance-based clustering*. In Figure 1.2, the cluster formed by the first three students has signature intervals $[89, 91]$ and $[88, 91]$ along the "language" and "music" dimensions respectively. The cluster in this simple example is an ideal one in that no other students got some scores within these signature regions. In this case, a single dimension (either language or music) is enough to unambiguously distinguish the cluster members. In reality, due to deviations from the ideal model and the presence of noise, cluster members can only be identified by cross-referencing a few relevant dimensions. For these imperfect clusters, each dimension can be assigned a separate *relevance* value to indicate how well it helps identify the cluster members.

When other kinds of similarity functions are used, the relevant dimensions could have other meanings. For example, in [68], a set of objects is similar if they have a coherent rise and fall pattern of projections across different dimensions. We refer to such a clustering process a *pattern-based clustering*. The relevant dimensions of a cluster correspond to the dimensions across which the objects exhibit similar patterns. Unlike distance-based clustering in which the relevance of each dimension can be evaluated individually, the relevant dimensions in pattern-based clustering must be identified in groups. In Chapter 3, we will discuss how a pattern-based clustering problem can be transformed to a distance-based one by adaptive transformation.

Before moving on, it is needed to emphasize the difference between projected clustering and feature selection. Although both concern the selection (and possibly construction) of important features, feature selection defines a feature space for the whole dataset, while projected clustering identifies a possibly different subspace for each cluster. Due to the difference, feature selection is performed prior to the actual data mining process[4], while subspace finding is performed

[4]Even in the wrapper model [44], the feature set is fixed before starting any run of the induction algorithm.

during the projected clustering process. Feature selection can be performed as a preprocessing step before projected clustering, but it alone cannot solve the projected clustering problem.

1.4 Gene Expression Profiles

This thesis does not only concern the technical issues of projected clustering, but also its specific application to gene expression data analysis. In this section we give a brief introduction to microarray technology and gene expression profiles.

A recent trend of genetics research has been moving from focusing on the functions of a particular gene to studying macroscopically the relationship between different genes in the whole genome. One technological breakthrough that leads to the trend is the invention of microarrays [62]. A microarray is a small chip (about one and a half inch wide) that contains an array of chemical reaction spots. The chemicals in each spots are designed to react with some different chemicals in the test samples. By using proper dyes, the amount of reacted chemicals in each spot can be quantified by measuring the light intensity at some specific frequencies.

Microarray technologies can be classified into a number of main types, including spotted arrays, short oligonucleotide arrays (Affymetrix GeneChips), long oligonucleotide arrays (Agilent), fibre optic arrays (Illumina) and serial analysis of gene expression (SAGE) [25]. Among them spotted arrays and Affymetrix GeneChips are the most widely used technologies. The former has the cDNA or oligos spotted on biochemically-treated solid surfaces such as glass and nylon, while the latter has the oligos synthesized *in situ* using photolithographic techniques.

One important application of microarrays is measuring the activity of different genes in a cell sample. During transcription, active genes produce messenger

RNA (mRNA) molecules that are complementary to one of the two strands of the double helix. Complementary DNA (cDNA) can be produced from the unstable mRNA molecules for measurement. DNA chips are designed in such a way that each spot contains a DNA sequence that can identify the cDNA molecules produced by a specified gene. When a cell sample is eluted on the surface of the array, the cDNA molecules of each gene will be bound to the complementary sequences of the corresponding spots. The quantity measured from each spot indicates the amount of cDNA produced, which in turn indicates the activity of the gene. The set of quantities obtained from the whole array is called an expression profile of the sample.

Depending on the technology employed, each quantity in a gene expression profile represents either the absolute expression level (e.g. Affymetrix GeneChips) or a relative expression ratio (e.g. cDNA microarrays). Due to the complex multi-step experimental procedures, gene expression profiles may contain missing and noise values. It is therefore a must to perform proper data cleaning, selection and transformation before carrying out any kind of data analysis.

In a typical gene expression experiment, the expression profiles of tens or even hundreds of samples are combined to form a large dataset, each measuring the activity of thousands of genes. The different samples may be taken from different kinds of cells (e.g. normal and tumor cells from the same patient), the same kind of cells subject to different external stimuli or taken at different time, etc. When performing analysis, a gene expression dataset is usually organized as a data matrix with the genes as the rows and the samples as the columns.

1.5 Clustering Gene Expression Profiles

Clustering is a popular data mining technique for extracting information from gene expression profiles. One major reason for this popularity is that clustering requires very little prior knowledge of the data, which is a big advantage for the application since the current knowledge on macroscopic gene interactions and pathways is still very limited.

We can link up some terms commonly used in this context to the ones introduced in Section 1.2. The rows of a gene expression dataset correspond to the *genes* (in some situations, they are more specifically referred to as *probes*, *expressed sequence tags (ESTs)* or *open-reading frames (ORFs)*, but we will not make the distinction here). Each column corresponds to a *sample* (*condition*, *tissue* or *time point*) and each projected value is called an *expression value*.

Interestingly, in gene expression data analysis, not only are clusters of genes meaningful, clusters of samples also have practical values in some applications. Whenever clustering is performed on samples, we will describe the clustering process as applying on the *transpose* of the dataset. In both cases, an object always refers to a row and a dimension always refers to a column in the resulting dataset. A gene, however, is represented by a row in the original dataset, but a column in the transposed dataset.

A large variety of traditional and novel clustering approaches has been used to generate many kinds of interesting clusters from gene expression profiles. Some recent studies include [13, 23, 26, 39, 40, 51, 63, 64]. The goal of these clustering methods is to partition similar objects (genes or samples) into clusters. Sample clustering is common in tumor studies for identifying tumor subtypes [8, 32, 52, 57]. Gene clustering has been used to predict groups of genes that have similar functions or are co-regulated [20, 30, 43]. It has also become very popular to cluster both samples and genes individually and visualize the results in a single figure [8].

All these approaches assume object similarity is measured in the input space formed by all the dimensions of a dataset. For example, when samples are being clustered, the similarity between two samples is based on the expression values of all the observing genes in the two samples. It has been pointed out that gene expression data may exhibit some checkerboard structures [47, 58]. Each block in the checkerboard is defined by a subset of genes and a subset of samples where the genes have similar expression patterns in the samples, which matches the definition of a projected cluster. The complexity and high dimensionality of gene expression datasets also make the existence of projected clusters highly probable. We are therefore interested in studying the feasibility of applying projected clustering on gene expression data.

1.6 Outline, Contributions and Scope of the Thesis

The remaining of this thesis is dedicated to the feasibility study. In Chapter 2 we review the previous works on projected clustering in the computer science and bioinformatics communities. Some clustering algorithms were developed specifically for analyzing gene expression profiles, while others are for general purpose. It is observed that many of the algorithms have a common potential problem in that they require users to input some hard-to-determine parameter values to assist the clustering process. The usefulness of the clustering results depends very much on the correctness of the parameter values being used. This is undesirable when working on gene expression profiles, since the datasets are formed by complex and mostly unknown biological processes, which makes the determination of correct parameter values extremely difficult.

In view of the problem, we will describe a new projected clustering algorithm in Chapter 3 that does not rely on user parameters. Obviously, this could never be achieved without making certain assumptions on the characteristics of clusters. We will show, however, that the assumptions being made by the algorithm are reasonable. In order to test the effectiveness and efficiency of the algorithm, we performed various kinds of experiments on both synthetic and real datasets, of which the results will be presented in Chapter 4. When presenting the results, some properties of the new clustering algorithm and some observed issues will also be discussed.

Chapter 5 is devoted to some overall discussions on the study, and to point out potential future works on the topic. Finally, Chapter 6 summarizes the whole thesis and draws the conclusions of the study.

The main contributions of this thesis are:

- Introducing a new projected clustering algorithm that does not rely on user parameters, which makes automatic analysis of a large amount of data with little domain knowledge feasible.
- Providing a rich set of experimental results on synthetic and real datasets, including some comparison results between various projected and non-projected algorithms under many different situations.
- Providing a thorough survey of the many projected clustering methods proposed in the computer science and bioinformatics communities.
- Studying the possibility of transforming a pattern-based clustering problem to a distance-based clustering problem, so that a single distance-based algorithm can handle both.

Although the ultimate goal of this study is to discover previously unknown relationships between different genes, this thesis concentrates on the technical issues related to the discovery of such relationships instead of the discoveries themselves. Interested parties are encouraged to try out the new algorithm in their own studies.

We have put the greatest effort in covering most existing projected clustering approaches in Chapter 2, but it is quite possible that some excellent approaches are still out of the list. Also, due to space limitation, we need to omit some details of each approach. Nonetheless, we believe the survey does have an adequate breadth and depth for the purpose of an overview of the topic.

In Chapter 3 we will describe the kind of projected clusters that our new algorithm tries to identify. We will then compare it mainly with the other algorithms with similar (or related) definitions of projected clusters. We make no attempts to claim that our algorithm performs better or equally well in all aspects as the other algorithms. The new algorithm does, however, successfully avoid a major usability problem that is common in most of the comparing algorithms. It also performs reasonably well in some other important aspects. We suggest to use the new algorithm as an automatic scanning of data to produce some initial clusters for inspection. More labor-intensive techniques can then be applied to extract deeper information from the datasets of which the clusters produced by the new algorithm appear to be interesting.

Chapter 2

Literature Review

In this chapter we review the previous studies on identifying clusters in subspaces. In Chapter 1, all these clusters are named "projected clusters". In the literature, different names have been given to these clusters, including *subspace clusters*, *projected clusters*, and *biclusters*. Each name is associated with a set of terms that describe the various concepts introduced in Chapter 1. For example, biclusters are usually described in terms of their "rows" and "columns" instead of their constituent "objects" and "relevant dimensions". For unity, we will stick to our preferred terms introduced in Chapter 1 as far as possible, and use other terms only when they give a much clearer meaning.

This chapter starts with three sections, featuring three closely related yet different research problems corresponding to the three names listed above: *subspace clustering*, *projected clustering* and *biclustering*. As the names imply, subspace clustering refers to finding clusters in subspaces, projected clustering refers to finding clusters that are projected onto some subspaces and biclustering refers to finding clusters constituted by both a subset of rows and a subset of columns. As far as we know, there exist no formal definitions of the three terms that reflect their differences. Based on our observations, we suggest to differentiate the three terms according to the following criteria:

- In subspace clustering and projected clustering, there is a primary clustering target. Subspace finding is merely to assist the identification of the relationship between different objects. For instance, in Figure 1.1, a cluster is clearly defined as a group of points (instead of a group of axis). Similarly, the clustering target in Figure 1.2 is the students instead of the subjects. In contrast, there is no primary clustering target in biclustering and both rows and columns are treated equally.
- In subspace clustering and projected clustering, the similarity between two objects is usually computed by a distance metric. A distance value is composed of the individual distance components along each dimension. This allows the relevance value of each dimension to be evaluated separately. In biclustering, there is a large variety of ways to calculate object similarity, most of which involve the rise and fall pattern of projected values. The relevance values of different dimensions usually depend on each other and cannot be evaluated individually.
- Subspace clustering algorithms search for and report all clusters that satisfy certain requirements, while most projected clustering and biclustering algorithms report only a small set of clusters that have the best quality.

This classification is not rigid in that exceptions can always be found. Nevertheless, it does reveal some fundamental options when defining clusters and clustering problems. It also provides a systematic organization for this chapter. As indicated by the title, this thesis focuses on projected clustering, which according to the above classification, has a primary clustering target, assumes a distance-based similarity, and produces a small set of high-quality clusters. However, pattern-based similarity and non-disjoint clusters are also desirable when working on certain types of gene expression profiles. We will discuss how a projected clustering algorithm can be extended to provide the functionality.

The last section of this chapter provides a brief summary of all the discussed problems and algorithms. It also motivates the development of a new algorithm by discussing some potential usability issues when applying the algorithms on real data.

2.1 Subspace Clustering

A subspace cluster is defined in a high-density region in the subspace formed by the relevant dimensions. The density is calculated as the number of objects per unit size of the region. As in all density-based clustering algorithms, a fundamental question to consider is the definition of "high" density. What should be the cutoff threshold for a region to be considered as dense? In this section we examine the answer of one clustering approach: bottom-up searching.

2.1.1 Bottom-up Searching Approach

In the bottom-up searching approach, the density threshold is supplied by user through an input parameter. The first proposed algorithm of this type is CLIQUE [5]. Initially each dimension is divided into units of the same width, and the number of objects projected onto each unit is counted. A unit is defined as dense if its object density exceeds the given threshold. Adjacent dense units are grouped to form one-dimensional clusters. Clusters form in different one-dimensional subspaces are not necessarily disjoint. A new iteration then starts to search for two-dimensional dense regions, which are regions whose projections on both constituting dimensions dense units. Adjacent two-dimensional dense regions again form clusters. The searching repeats for higher dimensionality until no more clusters can be formed.

The algorithm implicitly assumes that for a given cluster, the densities of objects along all its relevant dimensions are comparable. Once the unit width is fixed, dense units are unambiguously defined by the threshold parameter. The parameter is thus critical to the effectiveness of the algorithm.

Some variations of the algorithm have been proposed, including ENCLUS [17] and MAFIA [54]. These algorithms consider issues such as correlation between different dimensions, distribution of objects along each dimension, and variable unit width. The core part of the algorithms, however, is still a bottom-up searching according to a density threshold supplied by (or related to) a user parameter.

This approach is tightly related to fining frequent itemsets in association rule mining. Each unit on a dimension resembles an item, the density of objects corresponds to the support count, and a cluster is similar to a frequent itemset. This is why some approaches propose the use of association rule hypergraphs [35] to perform clustering. The tight relationship with frequent itemset mining unavoidably introduces a potential performance concern to the algorithms, namely the exponential growth of the number of subclusters as cluster dimensionality increases. By definition, the subspace clusters of CLIQUE obey the a priori property: if a set of objects form a dense unit in a d_I-dimensional space, they also form a dense unit in the $2^{d_I} - 1$ non-empty subspaces. This means the algorithm has an exponential time complexity with respect to cluster dimensionality. When working on transposed gene expression data where the dimensions correspond to the genes, the dimensionality of a cluster can be so large that causes the clustering algorithms to be very inefficient.

2.2 Projected Clustering

The definition of a projected cluster is very similar to that of a subspace cluster. A projected cluster is a group of objects with high similarity in the subspace formed by the relevant dimensions. In other words, when the objects are *projected* onto the subspace, their similarity becomes apparent. When similarity is measured by a distance function (which will be assumed in this section), a projected cluster is essentially a subspace cluster with high object density at some regions in the subspace formed by the relevant dimensions.

Unlike subspace clustering in which all regions that satisfy the density requirement are reported, projected clustering tries to find a number of disjoint clusters that optimize a certain evaluation function from all possible partitioning of objects and selections of relevant dimensions.

There are two major challenges in projected clustering that make it distinctive from traditional clustering. The first challenge is the simultaneous determination of both cluster members and relevant dimensions. Cluster members are determined by calculating object distances in the subspace formed by the relevant dimensions, while the relevant dimensions are determined by measuring the projected distances of the cluster members along different dimensions. One common approach to tackling this chicken-and-egg problem is to form some tentative clusters according to some heuristics, determine their relevant dimensions, and then refine the cluster members based on the selected dimensions. The heuristics being used are critical to the effectiveness of the algorithm. If inappropriate heuristics are used, the tentative clusters formed will not help the discovery of real clusters. We will discuss later how the performance of some existing algorithms may be affected by employing some heuristics that could be inappropriate in some situations.

The second challenge is the evaluation of cluster quality, which is in turn related to the determination of the dimensionality of each cluster. Traditionally, the quality of a cluster is measured by some objective functions. If a correct clustering model is chosen, a better objective score implies a larger chance that the clusters formed are correct. For example, as mentioned in the last chapter, k-means assumes that each cluster consists of a set of objects distributed closely around the centroid. The objective of the algorithm is thus to minimize the average within-cluster distance to centroid[1][2]:

$$W(\{C_I\}) = \frac{1}{k}\sum_{I=1}^{k} W_I \quad (2.1)$$

$$W_I = \frac{\sum_{x\in C_I}\sum_{v_j\in V}(x_j - x_{Ij})^2}{N_I} = \sum_{v_j\in V}\sigma_{Ij}^2, \quad (2.2)$$

where k is the number of clusters formed, x_j is the projected value of object x on dimension v_j, and x_{Ij} and σ_{Ij}^2 are the average and variance of projected values of the members of C_I along v_j. A straightforward generalization of W for projected clustering is as follows:

$$W^p(\{C_I\}) = \frac{1}{k}\sum_{I=1}^{k} W_I^p(V_I) \quad (2.3)$$

$$W_I^p(V_I) = \frac{\sum_{x\in C_I}\frac{1}{d_I}\sum_{v_j\in V_I}(x_j - x_{Ij})^2}{N_I} = \frac{1}{d_I}\sum_{v_j\in V_I}\sigma_{Ij}^2. \quad (2.4)$$

Similar objective functions are used in some previous studies [3, 4]. However, the function has a strong predilection for a small number of selected dimensions. Suppose the W^p score for a set of projected clusters is w, it is always possible to obtain an objective score not larger than w by deselecting some dimensions from some clusters. In other words, the optimal objective score is monotonically non-increasing as the clusters have fewer selected dimensions. To prove this property, consider a cluster C_I with d_I selected dimensions $v_1, v_2, ..., v_{d_I}$. Without loss of

[1]In this thesis, a capital letter, a small letter and a period in the subscript of a symbol indicate that the symbol describes a set of objects or values, a specific object or value, and all the objects or values in the dataset respectively. For example, x_{Ij} is the mean projected value of all members in cluster C_I along a specific dimension v_j while $x_{\cdot j}$ is the corresponding mean of all objects in the dataset.

[2]Some implementations prefer the non-averaged version of the objective score (i.e., without the normalization factors k in $W(\{C_I\})$ and N_i in W_I). The discussions in this section apply to both definitions.

generality, suppose $\sigma^2_{I1} \leq \sigma^2_{I2} \leq ... \leq \sigma^2_{Id_I}$. Now, by deselecting v_{d_I},

$$\begin{aligned}
W^p_I(V_I - \{v_{d_I}\}) &= \frac{1}{d_I - 1} \sum_{v_j \in V_I - \{v_{d_I}\}} \sigma^2_{Ij} \\
&= \frac{1}{d_I}(\frac{1}{d_I - 1} + 1) \sum_{v_j \in V_I - \{v_{d_I}\}} \sigma^2_{Ij} \\
&\leq \frac{1}{d_I}(\sigma^2_{Id_I} + \sum_{v_j \in V_I - \{v_{d_I}\}} \sigma^2_{Ij}) \\
&= \frac{1}{d_I} \sum_{v_j \in V_I} \sigma^2_{Ij} \\
&= W^p_I(V_I). \qquad (2.5)
\end{aligned}$$

As a result, if a clustering algorithm tries to optimize W^p, it would probably produce clusters each with only a few selected dimensions. In a real dataset, it is common to find a set of unrelated objects that have similar projected values along a few dimensions due to random chance. If the objects are treated as a cluster and the dimensions are selected as the only relevant dimensions, an excellent evaluation score will be resulted, yet the cluster is incorrect. The same argument holds for some other objective functions, such as the average between-cluster distance or the average within-cluster to between-cluster distance ratio. Algorithms that are based on the optimization of such functions would fail if they place no additional constraints on cluster dimensionality.

One possible solution is to get the dimensionalities of the clusters by some other means, and then optimize the objective function subject to the dimensionality requirements. The simplest way to obtain the cluster dimensionalities is to set them as algorithm parameters and request users to supply the values. While this solution has been adopted in some of the projected clustering algorithms, it has a usability implication.

Another solution is to design a new objective function for projected clustering. Summarizing the proposals of some previous studies[3, 4, 59], a projected cluster is likely to be correct if

1. Its selected dimensions have high relevance values (i.e., the average distances between the projections of the cluster members are small).
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 values of the selected dimensions are not due to random chance (a probability analysis considering the first two criteria can be found in Appendix A). 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. This solution may also affect the usability of the algorithms.

In summary, tentative clusters formation, quality evaluation and the determination of cluster dimensionalities are the major difficulties of projected clustering. We now examine how these problems are tackled in some proposed projected clustering approaches.

2.2.1 Hypercube Approach

In the hypercube approach DOC and its variant FastDOC [59], each cluster is defined as a hypercube with width 2ω, where ω is a user supplied value. 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 each 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 quality of them, which takes into account both the number of selected dimensions and the size of a cluster:

$$\mu(N_I, d_I) = N_I(\frac{1}{\beta})^{d_I}, \tag{2.6}$$

where β is another user parameter that defines the relative importance of the size and dimensionality of a cluster. The candidate cluster with the best evaluation score is accepted, and the whole process repeats to find other clusters.

It is proved in [59] that if a sufficiently large number of pivot points and random samples are tried, there is a high probability that a correct cluster will be formed. However, the number of trials can become very large for some parameter values. FastDOC sets an upper bound of the number of iterations to limit the maximum execution time, but the clustering accuracy is no longer guaranteed. In view of this, MineClus [72] makes use of efficient frequent-itemset discovery techniques to improve the time performance of the approach. Yet the accuracy of the algorithm still depends on the parameters ω and β in determining relevant dimensions and evaluating cluster quality.

2.2.2 Partitional Approach

Another approach is based on the traditional partitional clustering algorithms described in Chapter 1. PROCLUS [3] is one of the representative algorithms, which is based on the k-medoids method. As usual, some objects are initially chosen as the medoids, but before assigning every object in the dataset to the nearest medoid, each medoid is first temporarily assigned a set of "neighboring objects" that are close to it in the input space to form a tentative cluster. For each tentative cluster, the d input dimensions are sorted according to the average distance between the projected values 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 parameter value supplied by user. 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 regarded as outliers, which are replaced by some other objects to start a new iteration.

The objective function of PROCLUS is similar to the W^p score described before. As explained, the use of the objective function would cause PROCLUS to select too few relevant dimensions for each cluster if there are no restrictions on the number of selected dimensions. In order to tackle the problem, PROCLUS limits the average cluster dimensionality by the user parameter l. This may introduce a usability problem when working on high-dimensional datasets, where the number of possible l values is large and thus the correct value to use is hard to predict. Another potential problem arises when the real clusters have few relevant dimensions, in which case the cluster members may not be close to each other in the original input space. Since the tentative clusters are formed based on distance calculations in the input space, 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 a well mixture of objects from different real clusters.

Another partitional algorithm ORCLUS [4] was proposed to improve PROCLUS. Instead of drawing exactly k medoids at the beginning, more medoids are chosen to form more tentative clusters, which are later merged to form the final k clusters. In addition, ORCLUS selects the principal components (PCs) instead of input dimensions in order to detect arbitrarily oriented clusters. Principal component analysis (PCA) is performed on each cluster, but unlike traditional PCA where the goal is to identify PCs that capture a large proportion of total variance, the objective of ORCLUS is to find out PCs that have low average distances between projected values, which correspond to the PCs with small eigenvalues. As in PROCLUS, the number of PCs to be selected is governed by input parameters.

According to the experimental results reported in [4], ORCLUS is more accurate and stable than PROCLUS. Nevertheless, it still relies on user-supplied values in deciding the number of PCs to select. In addition, ORCLUS assumes that each cluster has the same number of relevant PCs, which seems to be quite unrealistic. Due to the heavy computation of PCA, the execution time of ORCLUS can also become intolerably long when working on high-dimensional datasets.

2.3 Biclustering

The third computational problem regarding the identification of clusters in subspaces is biclustering. Unlike the previous two problems, biclustering does not have a concrete technical definition. Biclustering is simply to cluster both rows and columns simultaneously, so that each resulting cluster consists of a subset of rows and a subset of columns. According to [18], the biclustering concept can be traced back to early 70's [37]. This section introduces a few different biclustering approaches, which all find applications in gene expression data analysis.

2.3.1 Minimum Mean Squared Residue Approach

The first approach assumes that each projected value in a cluster is the addition of three components: the background level, the row effect and the column effect. Figure 2.1 shows one such cluster with background level 5, row effects

Back.: 5	Column 0: 1	Column 1: 3	Column 2: 2
Row 0: 2	8	10	9
Row 1: 4	10	12	11
Row 2: 1	7	9	8

Figure 2.1. A bicluster based on the minimum mean squared residue approach.

$< 2, 4, 1 >$ and column effects $< 1, 3, 2 >$. For example, the value at row 0 and column 1 is calculated as 5 + 2 + 3 = 10.

The algorithms try to identify biclusters that have small deviations from the above perfect cluster model. The deviation is measured by the mean squared residue score. For a cluster C_I with relevant dimensions V_I, the score is defined as follows:

$$H_I = \frac{\sum_{x_i \in C_I, v_j \in V_I} (x_{ij} - x_{Ij} - x_{iJ} + x_{IJ})^2}{N_I d_I}, \tag{2.7}$$

where x_{Ij}, x_{iJ} and x_{IJ} are the column average, row average and block average respectively:

$$x_{Ij} = \frac{1}{N_I} \sum_{x_i \in C_I} x_{ij} \tag{2.8}$$

$$x_{iJ} = \frac{1}{d_I} \sum_{v_j \in V_I} x_{ij} \tag{2.9}$$

$$x_{IJ} = \frac{1}{N_I d_I} \sum_{x_i \in C_I, v_j \in V_I} x_{ij} \tag{2.10}$$

For a perfect cluster that has zero deviation from the model, the H score is zero. In terms of gene expression profiles, this occurs when all the genes of a cluster have exactly the same rise and fall pattern of expression across the relevant samples. In general, the smaller is the H score, the more similar are the expression patterns.

To identify the clusters, Cheng and Church [18] proposed a number of greedy algorithms to search for matrices with low H scores one after another. At the beginning a cluster is initialized to contain all objects in the dataset and with all dimensions selected. An iterative process then repeatedly adds to or removes from the cluster some rows and/or columns in order to decrease the H score of it. A cluster is accepted if the score drops below a user-defined residue threshold δ, or no more improvements can be made. In order to avoid the production of non-interesting clusters (clusters with extremely small sizes or clusters with constant projected values, which must receive good H scores), the algorithm also tries to maximize the cluster sizes and requires the clusters to have large row variances. After producing one cluster, the involved projected values are replaced by random numbers such that they create no structural influence to other clusters. It also prevents the same clusters from being reported multiple times. The searching process then repeats to form more clusters until a target number of clusters are formed.

In theory, the clusters produced by the Cheng and Church algorithms are not necessarily disjoint, but in reality due to the introduction of random numbers after discovering each cluster, it is difficult to identify clusters with substantial overlapping. This issue is addressed by the FLOC algorithm [69], which tries to locate all clusters at the same time and return them all together. The algorithm also takes care of missing values. Like the Cheng and Church algorithms, it also requires a residue threshold to define the stopping condition.

In [68], the pCluster model is defined to restrict the above model in a way that no object is allowed to have a trend across two dimensions that is very different from other objects. More specifically, for any two objects x and y in a cluster and any two relevant dimensions v_1 and v_2, the value

$$|(x_1 - x_2) - (y_1 - y_2)|$$

should not exceed a user-defined threshold δ. It is proved that if a matrix has this property, all its submatrixes also have this property.

Global background: 10

+

Cluster 1

Back.: 5	Column 0: 1	Column 1: 2	Column 2: ir.
Row 0: 2	8	9	0
Row 1: 3	9	10	0
Row 2: ir.	0	0	0

+

Cluster 2

Back.: 2	Column 0: 4	Column 1: 2	Column 2: 1
Row 0: 3	9	7	6
Row 1: ir.	0	0	0
Row 2: ir.	0	0	0

↓

Resulting dataset

27	26	16
19	20	10
10	10	10

Figure 2.2. A bicluster based on the plaid model.

The Plaid model [48] goes one step further to model the whole dataset as a superposition of clusters over the global background level. If a projected value belongs to multiple clusters, it will equal to the summation of all their background levels and the row and column effects. Figure 2.2 shows a dataset with two clusters that follows the Plaid model (ir means a row/column is irrelevant to a cluster).

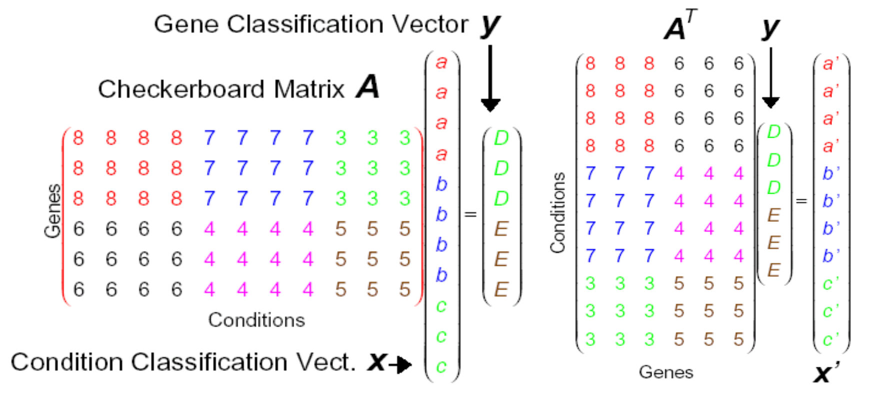
As in the Cheng and Church algorithms, clusters are discovered one after another by a greedy algorithm. When discovering each cluster, the effects of the previously discovered clusters are first removed, then the model parameters for the current cluster are estimated so that the deviation from a perfect cluster is minimized. The algorithm stops when the size of a cluster is not larger than a number of random clusters discovered from permuted data, or a target number of clusters has been formed.

The Plaid model is more suitable for identifying non-disjoint clusters, but the greedy nature of the algorithm may still discourage the overlapping of clusters. Suppose a projected value participates in multiple clusters. When identifying the first cluster, the projected value is a mixture of the effects of all the clusters. This means the value would appear to deviate greatly from the model of the cluster, which prohibits the inclusion of it into the cluster. The same situation happens for all remaining clusters, and the resulting clusters being discovered may have little overlap.

2.3.2 Spectral Approach

The spectral approach [47] adopts a model similar to the ones in the previous section in that each cluster is affected by a background level, row effects and column effects. But unlike the previous models, the spectral approach assumes that after normalization and reordering the rows and columns, a dataset becomes a checkerboard structure composed of aligned clusters. Figure 2.3 (from [47]) shows a data matrix $\mathbf{A}$ consisting of six perfect clusters where there is no row or column effects. The approach assumes that the row and column effects in real datasets can be eliminated by some normalization techniques, so that after reordering of the rows and columns the resulting dataset would consist of perfect clusters in the checkerboard format.

Suppose the conditions are classified according to the vector $\mathbf{x}$ and the genes are classified according to the vector $\mathbf{y}$, then $\mathbf{Ax} = \mathbf{y}$. Similarly, by taking the transpose of $\mathbf{A}$, $\mathbf{A^T y} = \mathbf{x}'$, where $\mathbf{x}' = \lambda^2 \mathbf{x}$ is a scalar multiple of $\mathbf{x}$. This results in an eigen problem $\mathbf{A^T A x} = \lambda^2 \mathbf{x}$. The solutions to the problem are the eigenvectors $\mathbf{x}$. If the scalar constants in $\mathbf{x}$ (i.e., a, b and c in Figure 2.3) form several clusters, the columns of $\mathbf{A}$ can be reordered accordingly. The row order can then be identified in a similar fashion.


Gene Classification Vector y
Checkerboard Matrix A
Genes
Conditions
Condition Classification Vect. x
A^T
y
Conditions
Genes
x'

(a) A normalized, row and column reordered gene expression matrix. (b) The transpose of the matrix.

Figure 2.3. The spectral approach.

The approach guarantees the detection of clusters if the data matrix after normalization can be reordered to exhibit a checkerboard structure. Yet it is not sure whether the structure does exist in real datasets. It seems too rigid to require all clusters to align in grids. The model also takes no account of outliers and irrelevant dimensions.

2.3.3 Order Preserving Submatrixes Approach

The above two biclustering approaches assume that all genes in a cluster have the same amount of response towards a certain condition. The order preserving submatrixes approach [12] employs a less stringent model. In this model, the absolute magnitude of response is unimportant. A submatrix is regarded as a cluster if the projected values in each row can be sorted in strictly increasing order by the same permutation of columns. A valid cluster is shown in Figure 2.4.

Each cluster is learnt from some (a, b) partial models, which specify only the first a and last b columns in the permutation. In Figure 2.4, the (1, 2) partial model specifies the permutation of the columns to be $< 3, ?, 1, 2 >$, where the symbol ? means the column has not been specified. Partial models with the same a and b values are compared according to their statistical significance.

	column 0	column 1	column 2	column 3
row 0	10	11	12	9
row 1	7	12	20	3
row 2	13	17	19	8
Permutation	2	3	4	1

Figure 2.4. An order preserving matrix.

Basically, a cluster is more favorable if there are more rows that support it, and a partial model is more likely to grow to a cluster with many supporting rows if the specified columns leave a big gap for unspecified columns. The OPSM algorithm proposed in [12] constructs (1, 1) partial models, keep the best l of them according to their significance (where l is a user parameter value), grows them to (2, 1) partial models, keep the best l of them, grows them to (2, 2) model, and so on, until l $(\lceil \frac{s}{2} \rceil, \lfloor \frac{s}{2} \rfloor)$ models are obtained, where s is also a parameter value.

The model adopted by the approach is more flexible than the previous two models, which is desirable since it is unlikely that a group of related genes will have exactly the same amount of response towards certain condition change. The model might, on the other hand, be too flexible in that it completely ignores the response magnitude. For instance, in Figure 2.4, row 0 is rather inactive while row 1 has significantly different projected values in different columns. It is not very intuitive to regard the two rows as of the same type. The model is also sensitive to noise, which can easily swap the sorting order of some projected values.

2.3.4 Maximum Weighted Subgraph Approach

All the above biclustering approaches define a cluster as a submatrix where the projected values of each row exhibits the same rise and fall pattern across the columns. The maximum weighted subgraph approach [65] has a different view of cluster. A dataset is viewed as a bipartite graph with the genes as one set of nodes and the samples as the other. A gene node and a sample node are connected if the expression level of the gene changes significantly in the sample (e.g. the absolute standard score of the projected value is larger than one), and the edge between the nodes is assigned a weight related to the significance of the value. A cluster is defined as a heavy biclique, where the weight of a subgraph is the total weight of the edges between every node pairs from different node sets. The weight of an edge is in turn related to the statistical significance of the projected value.

The algorithm, called SAMBA, guarantees to find k clusters with heaviest weights, where k is the target number of clusters. One constraint is that for each row node, there should be no more than a constant number of edges incident on it. Otherwise, the algorithm would have an exponential time complexity.

The approach suggests a new definition of cluster and brings new insights to future research directions. A potential limitation of the approach is the constraint on the number of incident edges of each node, which hinders the production of clusters that have large sizes or high dimensionalities.

2.3.5 Coupled Two-Way Clustering Approach

The last biclustering approach to be discussed is coupled two-way clustering [31], which is again very different from the previous approaches. This approach does not assume any fixed cluster model, but instead depends on the "plug-in" non-projected clustering algorithm to decide the type of clusters to form. The basic idea is to perform a series of non-projected clustering on a subset of genes and samples. The resulting clusters suggest new subsets of genes and samples to be attempted in later rounds of clustering.

More precisely, the algorithm keeps a pool G of gene sets and a pool S of sample sets. The initial pools contain the whole sets D (all genes) and V (all samples) respectively, and any other subsets of genes and samples that are believed to be meaningful according to some domain knowledge. The algorithm starts by taking an element from G and an element from S to form a data submatrix, and performs non-projected clustering on the submatrix and its transpose. The resulting gene and sample clusters are then put back to the two pools respectively. The clustering process repeats for all pairs of gene and sample sets each taking from the corresponding pool until no new clusters are produced.

The approach illustrates one possible way to produce projected clusters by non-projected clustering algorithms. However, only a specific kind of non-projected clustering algorithms can be used as the plugin. They should be able to automatically determine the number of clusters, and the number of distinct clusters produced should not grow uncontrollably. Otherwise, the algorithm will run for a very long time and produce an unmanageable amount of clusters.

2.4 Summary and Discussions

Table 2.1 summarizes some key properties of the approaches described in this chapter. Our clustering problem (Chapter 1) is most similar to the one of the partitional approaches. It is also a generalized version of hypercubes, since our definition does not require a cluster to have equal width along each relevant dimension. We will therefore focus on the algorithms from these groups.

We observe that most algorithms from these groups rely on some critical parameters to guide the clustering process, like the parameter l related to cluster dimensionality in PROCLUS and ORCLUS, the width parameter ω of hypercubes and the parameter β in the cluster evaluation function used by DOC, FastDOC and MineClus. It is not always possible for users to determine the best parameter values to use, especially when working on complex and high-dimensional gene expression profiles from which little domain knowledge is accessible. It would be more appropriate to have an algorithm that can intelligently determine the pa-

Approach	Algorithms	Cluster definition	Disjoint clusters	Clusters returned
• Bottom-up searching	CLIQUE, ENCLUS, MIFIA	High density region	No	All
• Hypercube	DOC, FastDOC, MineClus	High density hypercube	No	Best
• Partitional	PROCLUS, ORCLUS	Similar objects in the projected subspace	Yes	Best
• Minimum mean squared residue	Cheng and Church, FLOC, pCluster Lazzeroni and Owen	Objects with similar patterns	No	Best
• Spectral	Spectral	Constant value region	Yes	Best
• Order preserving Submatrixes	OPSM	Order-preserving submatrix	No	Best
• Maximum weighted subgraph	SAMBA	Region with significant expressions	No	Best
• Coupled two-way clustering	CTWC	Depending on plugin algorithm	No	All

Table 2.1. Summary of the reviewed clustering approaches.

rameter values to use on the fly according to the specific data characteristics of the dataset being clustered.

This motivates us to develop a new algorithm that learns the parameter values from data, which can be used to automatically analyze a lot of datasets without human intervention. In addition, we require the algorithm to be able to correctly identify clusters of extreme sizes and dimensionalities. It should not form incorrect tentative clusters, and should be scalable and insensitive to noise. The algorithm will be described in the next chapter.

Chapter 3

The HARP Algorithm

In this chapter we describe our new projected clustering algorithm HARP (a Hierarchical approach with Automatic Relevant dimension selection for Projected clustering) that satisfies the requirements stated in the last chapter. It is an agglomerative hierarchical clustering algorithm based on greedy merging of the most similar clusters. Three building components of the algorithm will be introduced first, followed by the complete algorithm and a complexity analysis of it. The last section of the chapter will be devoted to a discussion on some extensions of HARP, which facilitate pattern-based clustering and the production of non-disjoint clusters.

3.1 Relevance Index, Cluster Quality and Merge Score

HARP is classified as a projected clustering algorithm according to the classification in Chapter 2, which means it determines object similarity based on their distance in the projected subspace. In other words, given a cluster of objects, the relevance of a dimension in the cluster is related to the average distance between the projected values of the member objects on the dimension. In many previous studies [3, 4, 59], relevance is directly measured by this average distance. This

	Dimension A	Dimension B	Dimension C	Dimension 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

Figure 3.1. An example illustrating the idea of relevance.

may not be appropriate if the input dimensions have different ranges of values. Consider an example relation shown in Figure 3.1, where objects 1 and 2 form a 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. Obviously, both proposals are problematic as they do not satisfy the fundamental property of relevant dimensions: helping distinguish the cluster members from other objects. The projected values of objects 1 and 2 along the two dimensions do not form continuous intervals containing no other projected values. They cannot derive signatures of the cluster from the two dimensions. In comparison, dimensions A and B are actually more relevant to the cluster, even their absolute average within-cluster or between-cluster distances are worse.

From the example, it can be observed that if a dimension is relevant to a cluster, not only should the projected values of the cluster members be close to each other, they should also be well-separated from the projected values of other objects. This can be captured by a comparison of variance within the cluster and in the whole dataset. Recall that σ^2_{Ij} denotes the variance of projected values of all objects in C_I along v_j (the *local variance*) and denote $\sigma^2_{\cdot j}$ as the variance of projected values along v_j in the whole dataset (the *global variance*), the *relevance index* of v_j in cluster C_I is defined as follows:

$$R_{Ij} = 1 - \frac{\sigma^2_{Ij}}{\sigma^2_{\cdot j}}. \tag{3.1}$$

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 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 the stopping criteria of HARP.

To prevent the index from being undefined in some degenerating situations, we assume there does not exist any dimensions with zero global variances (on which all objects have the same projected values). If such a dimension does exist, it would not be useful at all and could 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.

Each of the local and global variances can be computed from a cluster feature (CF) [73], which consists of three additive components: the number of projected values, the sum of the values, and the sum of squares of the values:

$$\sigma_{Ij}^2 = \frac{\sum_{x_i \in C_I} x_{ij}^2}{N_I} - \left(\frac{\sum_{x_i \in C_I} x_{ij}}{N_I}\right)^2. \tag{3.2}$$

Whenever two clusters merge to form a new cluster, each CF of the new cluster can be readily computed by adding the three components of the corresponding CFs of the two original clusters separately. This makes the calculation of R very efficient.

In Figure 3.1, the R values of the four dimensions in 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.

Conceptually, incorporating the global variance in the relevance index is similar to performing standardization to the dataset. The use of the index thus implicitly performs standardization without the need of an explicit preprocessing step. An advantage of the index is the strong intuitive meaning of the sign of its values, which helps interpret the clustering results. The index also allows adaptive re-standardization of data after outlier removal. This is done by recalculating the global variances by subtracting the values of the outliers from the global CFs.

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:

$$Q_I = \sum_{v_j \in V_I} R_{Ij}. \tag{3.3}$$

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 (recall the three significance criteria of projected clusters discussed in Section 2.2. See also the analysis in Appendix A). 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 between two clusters. Basically, if two clusters can merge 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 merge to form a new cluster, the mean and variance of projected values will highly resemble the original values of the large cluster, and it 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 dimension v_j in C_{I_3} is defined as follows:

$$\begin{aligned} R^*_{I_3j} &= \frac{R_{I_1j|I_2} + R_{I_2j|I_1}}{2}, &\quad (3.4) \\ R_{I_1j|I_2} &= 1 - \frac{\sigma^2_{I_1j} + (x_{I_1j} - x_{I_2j})^2}{\sigma^2_{\cdot j}} \\ &= 1 - \frac{\sum_{x_i \in C_{I_1}} (x_{ij} - x_{I_2j})^2 / N_i}{\sigma^2_{\cdot j}}. &\quad (3.5) \end{aligned}$$

$R_{I_1j|I_2}$ is the adjusted relevance index of v_j in C_{I_1} given that C_{I_1} is merging with C_{I_2}. The numerator of its second term 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}. $R_{I_2j|I_1}$ is defined similarly. If the two clusters do not agree on the values along v_j, $(x_{I_1j} - x_{I_2j})^2$ will effectively diminish the R^* score of the dimension. With $R^*_{I_3j}$ defined, the merge score between clusters C_{I_1} and C_{I_2} can now be defined as follows:

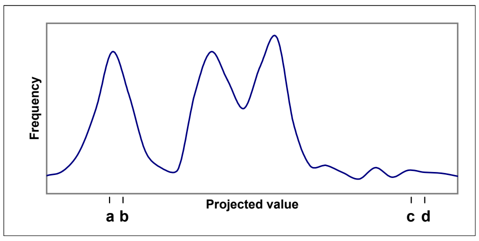
$$\begin{aligned} MS(C_{I_1}, C_{I_2}) &= \sum_{v_j \in V_{I_3}} R^*_{I_3j} \\ &= \sum_{v_j \in V_{I_3}} \frac{R_{I_1j|I_2} + R_{I_2j|I_1}}{2} \\ &= \sum_{v_j \in V_{I_3}} [1 - \frac{\sigma^2_{I_1j} + \sigma^2_{I_2j} + 2(x_{I_1j} - x_{I_2j})^2}{\sigma^2_{\cdot j}}]. &\quad (3.6) \end{aligned}$$

The MS score will be used to determine the merge order: merges with higher MS scores will be allowed to perform earlier.

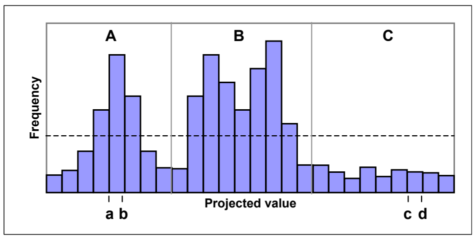
3.2 Validation of Similarity Scores

The MS function concerns both the quality and number of selected dimensions. A third criterion for evaluating cluster quality is the cluster size. Suppose there is a set C of objects all belonging to real clusters to which dimension v_j is irrelevant. If the size of C is small, it is not uncommon to find the projections of the objects in C on v_j being close to each other due to random chance. If C is large, the probability for the same phenomenon to occur is relatively smaller. Looking in another way, if a dimension has a high relevance index value in a cluster, 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 directly in the calculation of merge scores. However, it is possible to utilize the *potential* cluster size in estimating the significance of a cluster, which can be obtained from the frequency distribution of projected values. Figure 3.2a shows the distribution of the projected values of all data objects on a typical dimension that is relevant to some real clusters. The distribution contains a number of peaks, each corresponding to the signature of a real cluster. 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 merge with other clusters to form 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 score at the dimension, the cluster is unable to keep the high R value if it is to grow to a significant size. In other words, the high concentration of



(a) The frequency distribution.



(b) A histogram built from the distribution.

Figure 3.2. The frequency distribution of a typical dimension.

projected values is probably due to random chance. The R value of the cluster should therefore be invalidated in order to prevent more objects to merge into the cluster due to the fake signature.

The distribution inspires us to develop a histogram-based validation mechanism for preventing 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 3.2b). 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 the relevance index value of the dimension in the cluster will be rejected.

The validation mechanism contains two steps. First, the Kolmogorov-Smirnov goodness of fit test [16] is used to check if a dimension is irrelevant to all clusters, i.e., the distribution is essentially uniform. If the probability is high, the dimension will be removed from the dataset. The purpose of this step is to filter out dimensions where the peaks are caused by random fluctuations only. After filtering, each remaining dimension is expected to be relevant to at least one cluster. If a cluster C_I has mean x_{Ij} and variance σ_{Ij}^2 of projected values along dimension v_j, we check the mean frequency of the bins covering the range $[max(x_{Ij} - 2\sigma_{Ij}, min_{Ij}), min(x_{Ij} + 2\sigma_{Ij}, max_{Ij})]$, 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 while some abnormalities are ignored. To handle non-Gaussian cases, the minimum and maximum projected values are used to refine the boundaries of the range. When selecting the relevant dimensions of a cluster, if the mean frequency of the bins is below the mean of the whole frequency distribution, 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 rejected by the validation mechanism, v_j will make a zero contribution to the MS score.

We intentionally keep the validation mechanism simple in order to avoid introducing user parameters or computational overheads. It is the simplicity of the mechanism that makes it insensitive to the number of bins in the histogram. Any reasonable number can serve the purpose well, and we set it to $\sqrt{N}$ in all our experiments. We leave the more advanced uses of histograms in projected clustering to future research studies.

When working on gene expression datasets, the histogram-based validation is usually applied on gene clustering only, but not on sample clustering. This is because in the latter case the number of objects (samples) is usually too small to build a histogram that could simulate the real distribution of expression values.

3.3 Dynamic Threshold Loosening

When we introduced the MS function in Section 3.1 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 allowed to perform earlier. Practically, this is achieved by two internal thresholds R_{min} and d_{min}. Two clusters are allowed to merge if and only if the resulting cluster has d_{min} or more selected dimensions, and a dimension v_j is selected 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 their tightest (most restrictive, i.e., highest) values 1 and d respectively. All allowed merges produce clusters that contain identical objects, so the clusters must be correct. At some point, there will be no more qualified merges. The thresholds will be slightly loosened to qualify some new merges. Provided the loosening is mild, there is a high probability that the merges will produce correct clusters as the clusters are required to have a large number of selected dimensions with high R values (Appendix A). Whenever all qualified merges have been performed, the thresholds will be further 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 clearly differentiated from the irrelevant dimensions, which in turn ensures the formation of correct clusters.

In order to guarantee the minimum quality of the final 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 3.1, 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 scores much better than that of a random set of projected values.

There are many possible ways to loosen the threshold values. For example, from the last set of allowed merges, the average number of selected dimensions and their relevance index values can be computed to suggest which threshold has a greater loosening need. However, 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 3.4 for the details). The clustering accuracy of HARP is insensitive to the number of threshold levels, as we will show in Chapter 4. By default, we set it to the dataset dimensionality d such that after each threshold loosening, d_{min} is reduced by 1.

Note that by using the dynamic threshold loosening scheme, we do not require users to supply any parameter values for determining the relevant dimensions of the clusters.

3.4 The Complete Algorithm

With the core building blocks described in the previous sections, we now present the whole HARP algorithm. The skeleton of the whole algorithm is shown in Algorithm 3.1, and Procedures 3.2 to 3.6 list the pseudo codes of its main procedures.

At the beginning of the clustering process, each object forms a singleton cluster. The dimensionality and relevance thresholds d_{min} and R_{min} are initialized to their tightest values. For each cluster, the dimensions that satisfy the threshold requirements are selected. The merge score between each pair of clusters is then calculated. Only the merges that form a resulting cluster with d_{min}

or more selected dimensions are qualified. The other merges are being ignored.

```
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
```

Algorithm 3.1: The HARP algorithm.

```
Procedure BuildScoreCache($d_{min}$: dim. threshold,
$R_{min}$: rel. threshold)
1   Foreach cluster pair $C_{I_1}$, $C_{I_2}$ do {
2       $C_{I_3} := C_{I_1} \cup C_{I_2}$
3       SelectDimNew($C_{I_3}$, $R_{min}$)
4       If $d_{I_3} \geq d_{min}$
5           Insert $MS(C_{I_1}, C_{I_2})$ into score cache
6   }
End
```

procedure 3.2: The score cache building procedure.

The algorithm repeatedly performs the best merge according to the MS 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 back. The selected

```
Procedure SelectDim($C_I$: target cluster, $R_{min}$: rel. threshold)
1   Foreach dimension $v_j$
2       If $R_{Ij} \geq R_{min}$ and ValidRel($C_I, v_j$)
3           Select $v_j$ for $C_I$
End
```

procedure 3.3: The dimension selection procedure for an existing cluster.

```
Procedure SelectDimNew($C_{I_3}$: target cluster, $R_{min}$: rel. threshold)
1   Foreach dimension $v_j$ {
2       // $C_{I_3}$ is a potential cluster formed by merging $C_{I_1}$ and $C_{I_2}$
3       If $R^*_{Ij} \geq R_{min}$ and ValidRel($C_{I_1}, v_j$) and ValidRel($C_{I_2}, v_j$)
4           Select $v_j$ for $C_{I_3}$
5   }
End
```

procedure 3.4: The dimension selection procedure for a new cluster.

```
Procedure ValidRel($C_I$: target cluster, $v_j$: target dimension)
1   lowv := max($x_{Ij} - 2\sigma_{Ij}$, $min_{Ij}$)
2   highv := min($x_{Ij} + 2\sigma_{Ij}$, $max_{Ij}$)
3   If mean frequency of the bins covering [lowv, highv] <
    mean frequency of all bins
4       return false
5   Else
6       return true
End
```

procedure 3.5: The relevance index validation procedure.

```
Procedure UpdateScoreCache($C_{I_3}$: new cluster,
$d_{min}$: dim. threshold, $R_{min}$: rel. threshold)
1   // $C_{I_3}$ is formed by merging $C_{I_1}$ and $C_{I_2}$
2   Delete all entries involving $C_{I_1}$ and $C_{I_2}$ from cache
3   Foreach cluster $C_{I_4} \neq C_{I_3}$ do {
4       $C_{I_5} := C_{I_3} \cup C_{I_4}$
5       SelectDimNew($C_{I_5}$, $R_{min}$)
6       If $d_{I_5} \geq d_{min}$
7           Insert $MS(C_{I_3}, C_{I_4})$ into score cache
8   }
End
```

procedure 3.6: The score cache updating procedure.

dimensions of the new cluster are determined by its members according to R_{min}. According to the definition of R, if a dimension is originally not selected by both merging clusters, it must not be selected by the new cluster. However, 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.

We implemented three kinds of cache structures: priority queue (similar to the one used in [34]), quad tree, and Conga line [27]. Quad tree is optimal in access time, which takes $O(N)$ time per update[1], but it needs $O(N^2)$ space. Conga line is best for very large datasets as it takes only $O(N)$ space, but it needs $O(N \log N)$ time per insertion and $O(N \log^2 N)$ time per deletion. Priority queue takes worst case $O(N^2)$ space, but due to the two thresholds of HARP, usually only a small fraction of the $O(N^2)$ cluster pairs are allowed to merge, so the actual space being used is much lower. The time for each update is $O(N \log N)$. Depending on the memory available, HARP chooses the best cache structure to use and all structures give identical clustering results.

Whenever the cache becomes empty, there are no more qualified merges at the current threshold level. The thresholds will be loosened linearly according to the formulas in lines 2 and 3 of Algorithm 3.1. Further rounds of merging and threshold loosening will be carried out until a target number of clusters remains, or the thresholds reach their baseline values and no more qualified merges exist.

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.

[1]Here an update means an insertion or deletion of a cluster (instead of a merge score). When two clusters merge to form a new cluster, two deletions (of the original clusters) and one insertion (of the new cluster) are required.

The whole algorithm requires no user parameters in guiding dimension selection or merge score calculation, so it can easily be used in real applications. The high usability is attributed to the dynamic threshold loosening mechanism, which relies on the hierarchical nature of HARP. The parameter k that specifies the target number of clusters is optional. Like other hierarchical clustering methods, k can be set to 1 and the whole clustering process can be logged as a dendrogram[2], which allows users to determine the cluster boundaries from a graphical representation (e.g. [26]). These advantages justify the use of the hierarchical approach in spite of its intrinsic high time complexity. HARP is especially suitable for applications where accuracy is the first priority and the datasets are of moderate sizes, such as gene expression profiles. In order to deal with very large datasets, we will discuss some speedup methods in the next section.

Finally, we describe the outlier handling mechanism of HARP. It is similar to the one used by CURE [33] 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 merging of different real clusters due to the existence of noise clusters. As pointed out in [33], 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 non-outlier 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, this fill back process usually improves the accuracy.

[2]Due to the threshold requirements, it is not always possible to merge the objects into a single cluster at the end of clustering. In general, the dendrograms of HARP are forests of trees.

3.5 Complexity analysis

The time and space complexity of HARP depends on the cache structure used to store merge scores. At the beginning of each threshold level, building the cache requires $O(N^2)$ merge score calculations, each taking $O(d)$ time. In the worst case, no merges are qualified and the same order of building time is required for all $O(d)$ threshold levels, leading to $O(N^2d^2)$ cache building time. Whenever two clusters merge, all cached entries involving the clusters have to be deleted, the merge score between the new cluster and all other clusters calculated, and the new entries added to the cache. Suppose each cache access (insertion or deletion of a cluster) takes $O(f(N))$ time, then each merge requires $O(f(N) + Nd)$ time since $O(N)$ merge score calculations are required. Suppose $k \ll N$, then the total merging time involves $O(N)$ merges, which take $O(Nf(N) + N^2d)$ time. In total, the whole algorithm takes $O(N^2d^2 + Nf(N))$ time in the worst case, where $f(N)$ ranges from N (quad tree) to $N \log^2 N$ (Conga line).

While this theoretic worst case time complexity is quite daunting, the real execution time is much more encouraging as it is possible to optimize the performance of HARP in many ways. For example, for two clusters to be qualified for merging, the number of common dimensions that pass the histogram-based validation must exceed the d_{min} threshold. By checking the maximum number of such common dimensions of all cluster pairs, many threshold levels could be skipped if they contain no qualified 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, 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 as the merges must not be qualified.

In practice, the execution time of HARP is reasonable with medium-sized datasets, but it can become unacceptable when the dataset size or dimensionality is very large. We propose two ways to speedup the clustering process. When the dataset size is large, clustering can be performed on a random sample of objects. Upon completion of the clustering process, each unsampled object is filled back to the most similar cluster subject to the restriction of the final threshold values. When the dataset dimensionality is high, a constant number of threshold levels can be used (line 2 of Algorithm 3.1), so that the quadratic term with respect to d in the total time complexity becomes linear. We will show in the next chapter that these speedup methods reduce the clustering time dramatically with only minor impact to the clustering accuracy of HARP in our experiments.

The space requirement of HARP is dominated by the cache structure and the size of the dataset. The worst-case complexity is $O(Nd)$ when Conga line is used, and $O(N^2 + Nd)$ when quad tree or priority queue is used.

3.6 Extensions

Originated from traditional hierarchical clustering algorithms, HARP produces disjoint clusters and calculates object similarity (and relevance index values) based on the distance between projected values. When clustering gene expression profiles, it is sometimes more preferable to measure object similarity by the likeness of their rise and fall patterns of expression values. Two genes are regarded as similar if they have similar expression patterns, even there is a large absolute difference between their expression values. It is also desirable to allow clusters to be non-disjoint when producing gene clusters since a single gene may be involved in multiple functions.

HARP can be extended to provide the functionality. To perform pattern-based clustering, the row effects (Section 2.3.1) of the clusters have to be removed. Consider the perfect bicluster shown in Figure 2.1. If the row effects are removed, the cluster will become the one shown in Figure 3.3. At each relevant dimension,

Back.: 5	Column 0: 1	Column 1: 3	Column 2: 2
Row 0: 0	6	8	7
Row 1: 0	6	8	7
Row 2: 0	6	8	7

Figure 3.3. The bicluster shown in Figure 2.1 with the row effects removed.

all members have exactly the same projected value. In reality, when clusters are imperfect, the projected values on each relevant dimension will concentrate on an exceptionally small range. This means the dimensions will receive high relevance index values in the cluster, and HARP will be able to identify the high quality of the cluster by the Q and MS functions even the original projected values of the objects are quite different.

If the real relevant dimensions are known, the row effects can be removed by deducting each projected value by the mean of its row over the relevant dimensions (mean centering). Each projected value x_{ij} in cluster C_I is transformed as follows:

$$x'_{ij} = x_{ij} - x_{iJ} \tag{3.7}$$

It can be easily verified that mean centering can effectively remove the row effects of the dataset in Figure 2.1. Although the resulting data values are different from the ones shown in Figure 3.3, the rise and fall patterns in the original cluster can be captured by the distance between objects in both cases. In terms of gene expression, the transformed value represents the expression level of gene i in sample j relative to its average expression in the relevant samples.

Suppose a cluster contains objects all from the same real cluster, it would be ideal to perform mean centering according to the real relevant dimensions of the real cluster. Unfortunately, the real relevant dimensions are unknown during the clustering process until most members of the real cluster have been identified. To tackle the problem, the selected dimensions of each cluster are used to approximate the real relevant dimensions, and a new mean centering is performed on the members of a cluster every time the selected dimensions of it are updated after it merges with another cluster.

A problem arises when we want to calculate the merge score between two clusters. The merge score of HARP (MS) is a summation of relevance index values of the selected dimensions of the resulting cluster plus a penalty term. In order to determine the selected dimensions of the resulting cluster, we need to compare the relevance index value of each dimension with the R_{min} threshold. The relevance index values can be computed accurately only when the mean centering has been performed. However, the mean centering can be performed only when the set of selected dimensions is defined.

We use a simple heuristic to break this infinite looping. The selected dimensions of the new cluster are estimated by the intersection of the two sets of selected dimensions of the merging clusters, since the chance for a dimension to be selected for the new cluster is much higher if it is originally selected by both merging clusters. This set of temporarily selected dimensions is used to perform mean centering on the members of the two clusters. The relevance index values of each dimension of the new cluster can then be computed, and a refined set of selected dimensions can be obtained by selecting all dimensions with R^* exceeding R_{min}. This refined set can again be used to perform another round of mean centering and dimension selection. The elements in the set of selected dimensions usually become stable after a few rounds of refinement, at which the merge score between the clusters will be determined.

To produce non-disjoint clusters, each object is allowed to join multiple mature clusters after their structures have been identified during the normal clustering process. More specifically, when normal clustering completes, for each produced cluster C_I, all the objects in the dataset will be examined to see if they can be merged into C_I without lowering its quality. Each object is regarded as a singleton cluster, and its projected values are mean centered according to the selected dimensions of C_I. When calculating the MS score between a singleton cluster and C_I, d_{min} and R_{min} are set as the number and minimum R value of the selected dimensions of C_I, which prevents the quality of C_I from being degraded. All the objects involved in the qualified merges become members of C_I. Since each object is allowed to join multiple clusters, the final clusters are not necessarily disjoint.

Chapter 4

Experiments and Discussions

In this chapter we report various experimental results of HARP and some other algorithms in comparison. The first section covers the methods and procedures of the experiments, and the second section covers the experimental results and some discussions.

4.1 Methods and Procedures

4.1.1 Datasets

We performed experiments on both synthetic and real datasets. Synthetic datasets were used to test the capability of HARP in dealing with various data characteristics, while real datasets were used to verify the practicality of HARP in handling complex real data.

4.1.1.1 Synthetic Data

Table 4.1 lists the default parameters used in synthetic data generation.

When generating a dataset, the size of each cluster and the domain of each

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^r)	$\frac{d}{k}$ 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 error rate (e)	5%
Artificial outlier rate (o)	0%

Table 4.1. Data parameters of the synthetic datasets.

dimension were first determined randomly according to the data parameters. Having different cluster sizes creates different peak heights at the frequency distributions, which tests the stability of the histogram-based validation mechanism. The different domain sizes are to test the importance of the standardization factor in 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 study, 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 followed the signature, a projected value would be generated from the Gaussian distribution. Otherwise, and for all irrelevant dimensions, the values would be generated from a uniform distribution over the whole domain.

The default dataset size and dimensionality values (500 and 20) are moderate in order to allow the extensive experiments to be run in a reasonable time. We also produced some large datasets to test the scalability of HARP. The results will be presented in a later section.

4.1.1.2 Real Data

We performed clustering on a number of real datasets of different types:

Lymphoma: It is a dataset used in studying distinct types of diffuse large B-cell lymphoma (DLBCL)(Figure 1 of [8]). It contains 96 samples, each with 4026 expression values. The samples are categorized into 9 classes according to the category of mRNA sample studied. We worked on the transposed dataset with the genes as the input dimensions, and used HARP to perform distance-based clustering to produce 9 sample clusters. The transposed dataset was standardized so that each gene has a zero mean and a unit variance of expression values. This standardization is not necessary for HARP due to its use of relevance index. It was performed merely to allow fair comparisons between the results produced by different algorithms, as to be explained in Section 4.1.6. Each relevant dimension of a cluster represents a gene that has similar expression levels in the member samples of the cluster, which is a potential signature of the samples.

Leukemia: It consists of 38 bone marrow samples obtained from acute leukemia patients [32], 27 of them were diagnosed as ALL (acute lymphoblastic leukemia) and 11 as AML (acute myeloid leukemia). Each sample is described by the expression values of 7129 genes. The ALL samples can be further classified into two classes, one containing 19 B-cell ALL samples (B-ALL), and the other containing 8 T-cell ALL samples (T-ALL). In [32], the 38 samples are partitioned into two and four clusters by self organizing maps. In the former case, the two clusters clearly correspond to ALL and AML samples respectively, with only 4 samples being put into a wrong cluster. In the latter case, AML and T-ALL each occupies a cluster, and B-ALL occupies the other two. Only two samples are put into a wrong cluster. We used HARP to perform distance-based clustering on the transposed dataset to form two sample clusters, and compare the results with the ones presented in [32].

Yeast: The original dataset was published in [20]. It contains the expression levels of 6,218 yeast ORFs at 17 time points taken at 10 minute intervals, which cover nearly two full cell cycles. The dataset used here is the subset selected according to [66] that contains 2,884 genes. We preprocessed the data according to the method suggested in [18], and used HARP to perform pattern-based clustering to produce non-disjoint gene clusters using the two extensions. As in [18], we treated two genes as similar if they have complementary expression patterns in the corresponding subspace, i.e., the two genes constantly show opposite rise and fall patterns across the relevant dimensions. This is accomplished by having two copies of each gene in the dataset, one with the original expression values, and the other the negation of them. This results in two nearly identical copies of every cluster being formed. In the results reporting in the coming sections, all duplicated clusters and duplicated genes in a cluster are removed.

Food: We also used a food dataset to explore the possible application of HARP in other domains. It contains the weights and 6 attributes (Fat, Food Energy, Carbohydrate, Protein, Cholesterol and Saturated Fat) of 961 food items[1]. We followed [48] to normalize the six attributes by the weight, and standardized each column to have unit standard deviation. Since the dataset contains no known class labels, we treat the clustering as an exploratory task and report some interesting findings. We choose to report the results of this dataset because HARP was able to discover some interesting clusters that could not be found by a non-projected clustering algorithm.

4.1.2 Comparing Algorithms

To demonstrate the capability of HARP, we compared it with various projected and non-projected algorithms. For the synthetic datasets, we chose PRO-

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

CLUS [3], ORCLUS [4] and FastDOC [59] as the representatives of projected algorithms as they have reasonable worst case time complexity 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 [55] and KPrototype [41] (based on k-medoids and k-means respectively), and CAST [13], an algorithm designed for clustering high-dimensional gene expression profiles. We believe our choice of algorithms covers a wide spectrum of clustering approaches.

For the real datasets, we mainly compared the results of HARP with those reported in the corresponding references.

4.1.3 Similarity Functions

We used MS as the merge score of HARP, and Euclidean distance as the similarity function of all other projected algorithms as all of them adopt a distance-based relevance definition. For non-projected algorithms, we used both Euclidean distance and Pearson correlation to measure object similarity. CAST can only work with similarity functions that have a finite range, so only Pearson correlation (with range [-1, 1]) was used.

4.1.3.1 Algorithm Parameters

Some of the comparing algorithms require the input of some parameter values. We compare the performance (accuracy, noise tolerance, etc.) of HARP with the other algorithms in two aspects:

- The peak performance when correct parameter values are inputted to the

Algorithm	Parameter	Values used
CAST	t	0.05 to 0.5, 10 values
CLARANS	maxneighbor	250
	numlocal	5
FastDOC	α	$\frac{1}{2k}$
	β	0.1 to 0.25, 4 values
	ω	0.02 to 0.1, 5 values
	d_0	d
	MAXITER	100000
ORCLUS	α	0.5
	k_0	15 k
	l	10% to 90% of d, 9 values
PROCLUS	A	20
	B	5
	l	10% to 90% of d, 9 values

Table 4.2. Parameter values used in the experiments.

required algorithms, which corresponds to situations where a lot of domain knowledge is accessible.

- The average performance when a number of reasonable parameter values are used, and the performance degradation as the inputs deviate from the correct values. This is to test the applicability of the algorithms in real situations where only a little domain knowledge is available.

In all the experiments the target number of clusters was set to the number of real clusters if it was known. For each of the other parameters, various reasonable values were tried. Table 4.2 lists the user parameters of the algorithms and the values used. HARP, non-projected hierarchical and KPrototype require no user parameters except k.

CAST and FastDOC produced the desired number of clusters only at some specific parameter values. All results that contain fewer than the desired number of clusters were discarded.

4.1.4 Execution

Except HARP and the non-projected hierarchical method, all other algorithms do not give deterministic results, i.e., different runs on the same dataset may give different results. To avoid random bias, we repeated each experiment of the algorithms five times. For each repeated run, only the result that has the best algorithm-specific criterion score will be considered in the discussions below.

4.1.5 Evaluation Criteria

We used the Adjusted Rand Index (ARI) [70] as the performance metric for clustering accuracy when all members of the real clusters are known. It is based on the Rand Index [61], which measures how similar are the partition of objects according to the real clusters (U^r) and the partitioning in a clustering result (U^c). Denote a, b, c and d as the number of object pairs that are in the same cluster in both U^r and U^c, in the same cluster in U^r but not U^c, in the same cluster in U^c but not U^r, and in different clusters in both U^r and U^c respectively, Rand Index is defined as follows:

$$Rand(U^r, U^c) = \frac{a+d}{a+b+c+d}. \tag{4.1}$$

ARI modifies the Rand Index by taking into account the expected index value of a random partitioning. It has the following formula:

$$ARI(U^r, U^c) = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)} \tag{4.2}$$

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

Another important evaluation criterion of projected clustering algorithms is the accuracy of dimension selection. 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 total number of selected dimensions. Recall is the number of real relevant dimensions being selected divided by the actual number of real relevant dimensions of the majority class. The reported value of a clustering result is the average of all the clusters.

We also measured the importance of subspace finding in the analysis of real datasets by calculating the change of within-cluster and between-cluster distances due to dimension selection. For each projected cluster, we computed the following three distance ratios:

$$A_1(C_I) = \frac{\sum_{x_i \in C_I, v_j \in V_I} (x_{ij} - x_{Ij})^2 / d_I}{\sum_{x_i \in C_I, v_j \in V} (x_{ij} - x_{Ij})^2 / d} \tag{4.3}$$

$$A_2(C_I) = \frac{\sum_{x_i \in C_I, v_j \notin V_I} (x_{ij} - x_{Ij})^2 / (d - d_I)}{\sum_{x_i \in C_I, v_j \in V} (x_{ij} - x_{Ij})^2 / d} \tag{4.4}$$

$$A_3(C_I) = \frac{\sum_{x_i \notin C_I, v_j \in V_I} (x_{ij} - x_{Ij})^2 / d_I}{\sum_{x_i \notin C_I, v_j \in V} (x_{ij} - x_{Ij})^2 / d} \tag{4.5}$$

A_1 measures the increase in compactness of the cluster due to dimension selection, A_2 measures how irrelevant are the non-selected dimensions, and A_3 measures the increase in separation of the cluster members from other objects due to the selection. For a good cluster, A_1 should be smaller than one, A_2 should be greater than one, and A_3 should be larger than A_1.

For pattern-based clustering, we will use the mean squared residue score H introduced in Section 2.3.1 to evaluate the quality of clusters. A smaller H score indicates a less severe deviation from the perfect cluster model, which indicates a better cluster. Obviously, clusters with smaller sizes are more likely to receive small H scores. We will therefore augment the comparison results with the sizes of the clusters.

Algorithm	Without standardization	With standardization
FastDOC	0.78	1.00
HARP	1.00	1.00
ORCLUS	0.72	0.98
PROCLUS	0.86	0.94

Table 4.3. ARI values of the clustering results of the projected algorithms with and without standardization.

4.1.6 Data Preprocessing

When introducing the relevance index, we claimed that the variation of global variance across different dimensions could mislead the dimension selection mechanism. To verify this claim, we generated an "easy-to-cluster" dataset with $l^r = 12$ and $o = 0$. We tested the clustering accuracy of the projected algorithms FastDOC, HARP, ORCLUS and PROCLUS with and without standardizing the values of each dimension, using correct user parameter values (for FastDOC, various parameter values were tried and the best result is reported). Table 4.3 shows the ARI values of the results.

With the global variance taking into account in the R index, the performance of HARP is invariant to the standardization process. For all the other methods, the clustering accuracy was improved by standardization. This confirms the importance of the standardization factor in R. For fair comparisons, all the synthetic datasets used in the coming sections were standardized.

4.1.7 Outlier Handling

From the preliminary experiment for studying the importance of data standardization, we noticed that FastDOC tends to discard an unnecessarily large amount of outliers. For the two reported results, 67% and 31% of objects were left unclustered, even the dataset contains no artificial outliers. According to [4] and our other experimental results, PROCLUS also has a similar problem in some situations. 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.

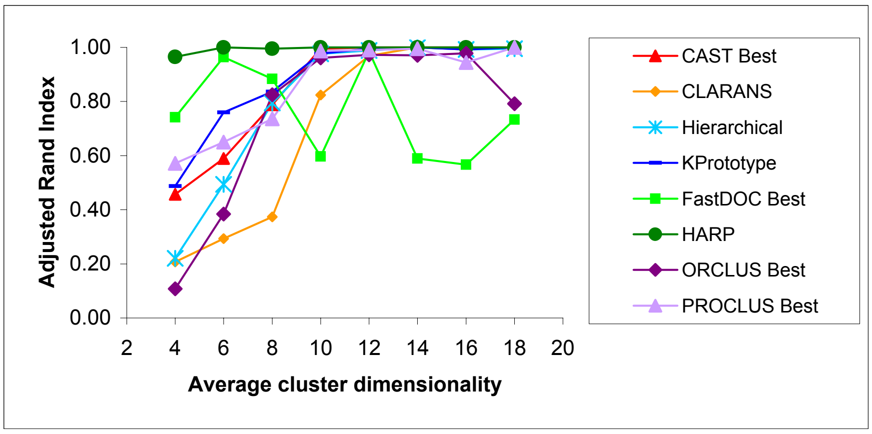
4.2 Results and Discussions

4.2.1 Results on Synthetic Data

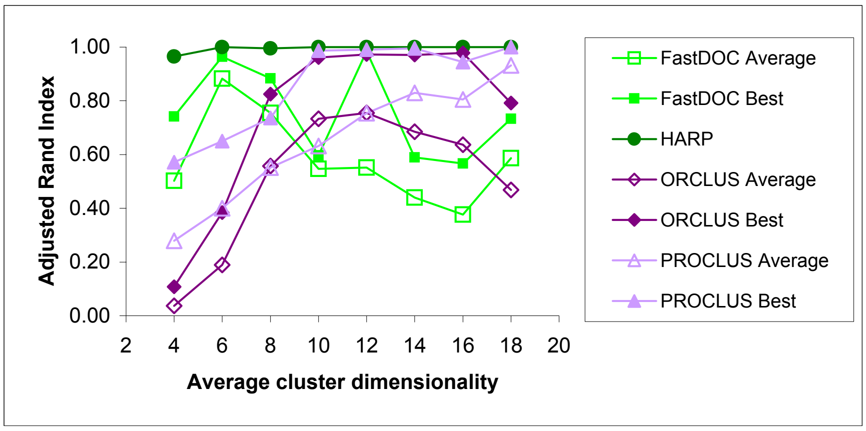
4.2.1.1 Clustering Accuracy

The first set of experiments concerns how the clustering accuracy is affected by cluster dimensionality l^r. We generated eight datasets with l^r ranging from 4 to 18, which account for 20% to 90% of the dataset dimensionality. For clarity, we present the results in four different charts in Figure 4.1 and Figure 4.2. In the charts, and in the other figures to be presented later, lines labeled "best" and "average" represent the results of an algorithm with the highest ARI values and the average result obtained by trying all the parameter values specified in Table 4.2 respectively. Since CLARANS, HARP, Hierarchical and KPrototype did not need to try on multiple sets of parameter values, only one line is presented for each of them.

Figure 4.1a shows the best results of the algorithms. Most algorithms were highly accurate at large l^r values, but for l^r values lower than 50% of d, the performance difference between them became apparent. HARP got the highest ARI values among all the algorithms on all the datasets, and remained highly accurate even each cluster had 80% of the dimensions being irrelevant to them.

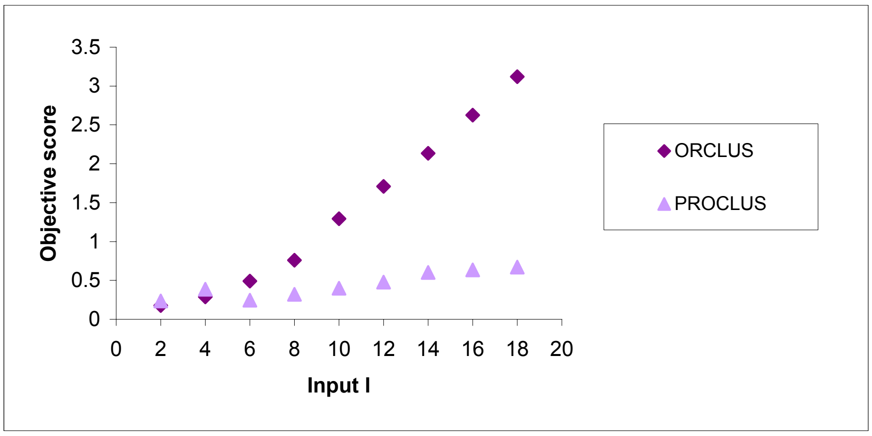


(a) The results with the highest ARI values of each algorithm.

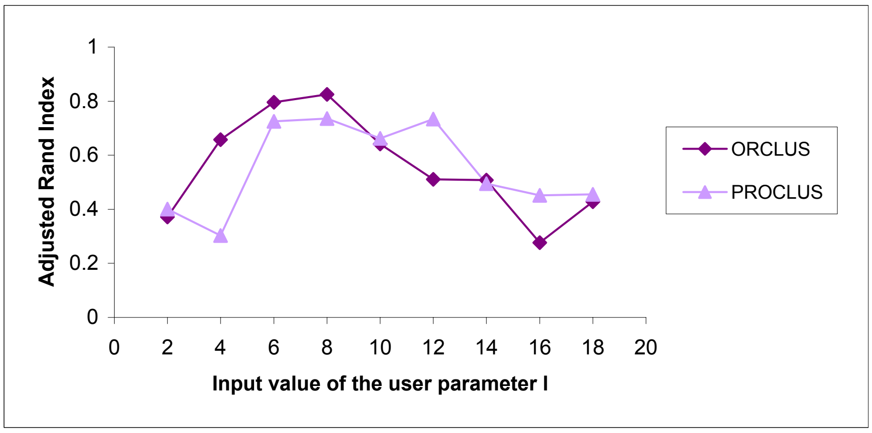


(b) Comparing the results with the highest ARI values with the average results of the projected clustering algorithms using different parameter values.

Figure 4.1. Clustering results on datasets with different cluster dimensionalities.



(a) Objective scores of the results of PROCLUS and ORCLUS.



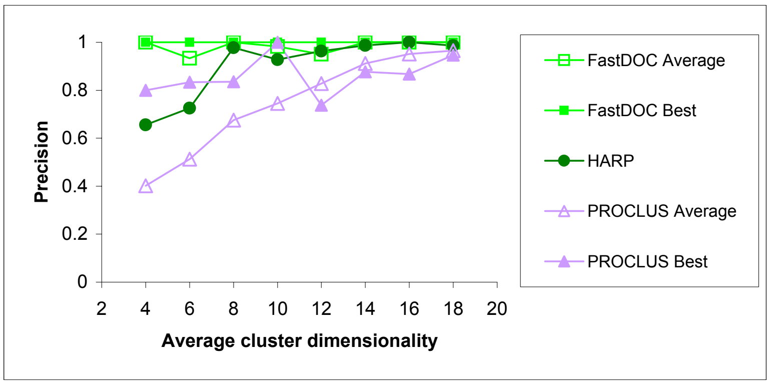
(b) Clustering accuracy of PROCLUS and ORCLUS.

Figure 4.2. Clustering results on the dataset with $l^r = 8$, using various user parameter inputs.

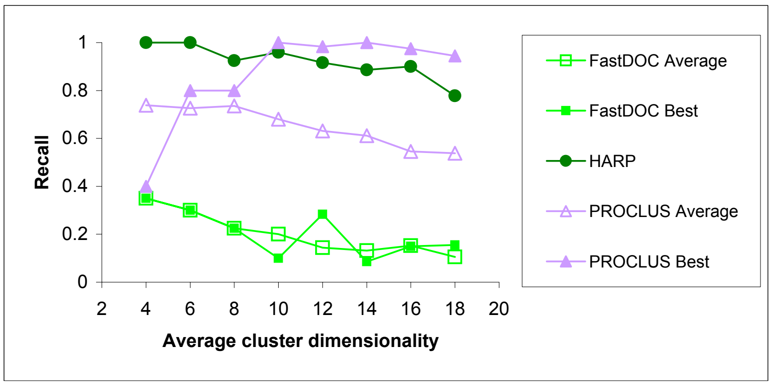
The results of ORCLUS reported in [4] are better than the current results, which is likely caused by the small sizes of our synthetic datasets since ORCLUS works well on large datasets that contain sufficient values for performing PCA, but its performance on small datasets is less competitive. FastDOC 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^r values, but some good results were due to the correct input of parameter values. Figure 4.1b compares the best and average results of FastDOC, ORCLUS and PROCLUS after trying different parameter values. The average results have much lower ARI values than the best results, which means when incorrect parameter values are used, the performance of the algorithms could be greatly affected. In many applications, it is hard for users to know the correct parameter values. Furthermore, as explained before, the objective scores of the results may not be useful in choosing the best parameter values to use, since they may bias towards clusters with low dimensionalities (Figure 4.2a). This means in real situations if the correct parameter values are unknown, the optimal results can hardly be obtained. Figure 4.2b shows the typical fluctuation of accuracy of PROCLUS and ORCLUS with various parameter inputs, taken from the results on the dataset with $l^r = 8$. 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. For HARP, the accuracy is independent of user inputs.

From Figure 4.1b, it is also noted that even supplied with correct parameter values, PROCLUS and ORCLUS were unable to achieve very good accuracy at small l^r values. 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 avoid forming incorrect tentative clusters during



(a) Precision of the selected dimensions.



(b) Recall of the selected dimensions.

Figure 4.3. Accuracy of the selected dimensions of the results produced by FastDOC, HARP and PROCLUS.

the early stage of clustering.

Next we investigate the selected dimensions of the projected clustering algorithms. Figures 4.3a and 4.3b show the average precision and recall of the selected dimensions of the results produced by FastDOC, HARP and PROCLUS.

Interestingly, HARP behaved differently at different l^r values. For clusters with high dimensionalities, HARP tended to be conservative in dimension selection as reflected by the high precision and relatively low recall. This means HARP deliberately avoided selecting irrelevant dimensions when the selected ones were enough for identifying cluster members correctly. However, at low l^r

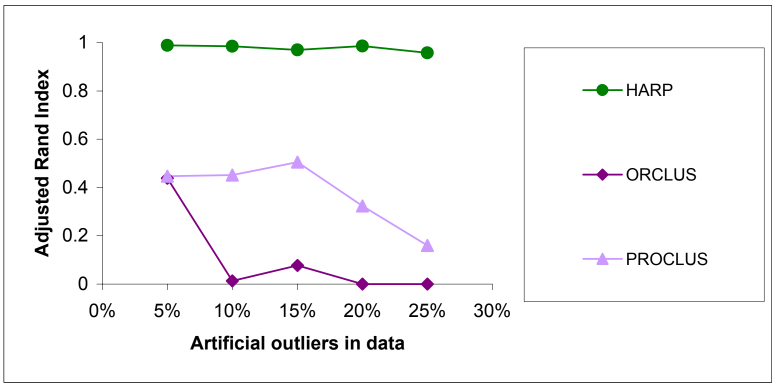
values, HARP tried to include all relevant dimensions in order not to miss any useful information, with the expense of also selecting some irrelevant dimensions. We argue that this is acceptable as recall is more important than precision when l^r is small since missing a single relevant dimension may mean missing a substantial proportion of information, while including a few irrelevant 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. 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.

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

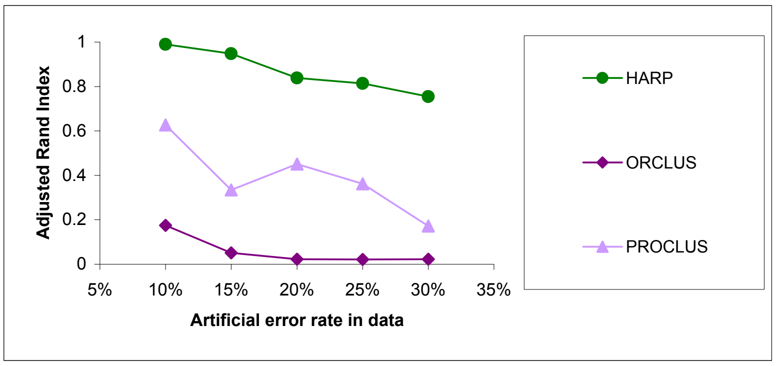
4.2.1.2 Imperfect Datasets

Although the above experiments show that HARP is highly accurate and usable, the synthetic datasets used are too ideal with no outliers ($o = 0$), a low error rate ($e = 5\%$) and clear signatures ($\sigma_{Ij} = 3 - 5\%$ of domain). In the coming experiments we demonstrate the influence of these data parameters on the clustering accuracy.

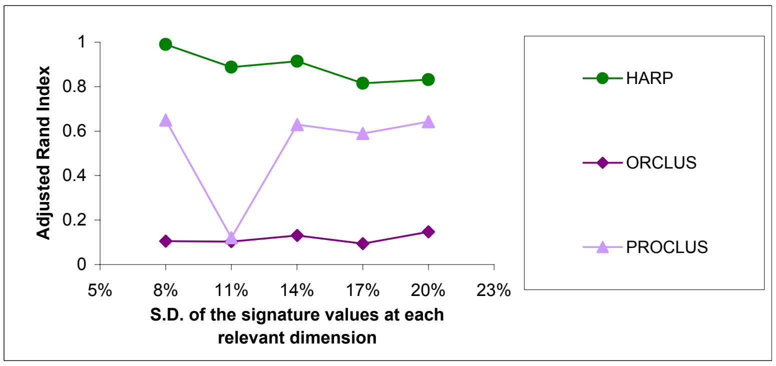
We fix l^r to 6 (30% of d) since at this value the performance difference between projected and non-projected algorithms becomes clear. We generated three sets of data, with increasing o, e and σ_{Ij} respectively. We tested the performance of HARP, using PROCLUS and ORCLUS (with correct parameters supplied) as reference. The results are shown in Figure 4.4.



(a) Clustering accuracy with the presence of artificial outliers.



(b) Clustering accuracy with the presence of artificial errors.



(c) Clustering accuracy with various spread of signature values.

Figure 4.4. Clustering results of HARP, ORCLUS and PROCLUS on imperfect data.

Figures 4.4a shows the ARI values of the algorithms with the presence of outliers. HARP remained highly accurate, and the amount of objects discarded by the outlier handling mechanism was found to highly resemble the actual amount of outliers with only a little over-discarding. In comparison, ORCLUS and PROCLUS discarded more objects and had unsatisfactory clustering accuracies. ORCLUS appears to be very sensitive to outliers, which may due to the fact that in latter iterations ORCLUS picks the cluster seeds from the centroids of the cluster. When the clustering accuracy is low, each cluster consists of objects from many different real clusters and the centroids will be a mixture of their signatures. As a result, the centroids will be similar to each other, but dissimilar to any object in the dataset. This phenomenon ruins the outlier removal mechanism of ORCLUS (which removes objects that have a longer projected distance from the seed of its assigned cluster than the projected distance between the seed and its closest seed), causing it to discard a substantial amount of objects.

Figure 4.4b shows the results with increasing amount of data errors. The figure shows that the accuracies of all three algorithms went down as more errors were introduced, but HARP only had a mild deterioration. Similarly, Figure 4.4c shows that as the cluster signatures became less focused, HARP only had a gentle decrease in accuracy. The sudden drop of accuracy of PROCLUS at 11% was due to a biased selection of medoids.

4.2.1.3 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 4.5. Part a shows that the accuracy of HARP was unaffected by the dataset size, and part b 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. When the time used in repeated runs for avoiding random bias and trying different parameter values is also included, the execution time of HARP was also comparable to PROCLUS. We also tested if the sample-based speedup technique described in Section 3.5 is feasible. Clustering was performed on the dataset with 10000 objects using various sample sizes. From the results shown in Figure 4.5c, for reasonable sample sizes, the execution time was much improved with only a little impact on the accuracy.

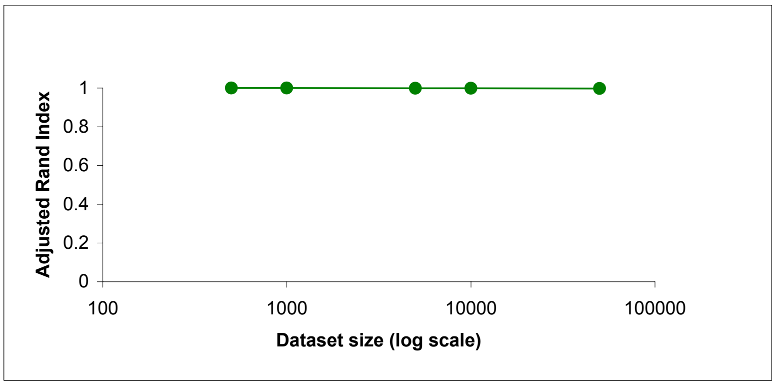
The results with increasing dataset dimensionality are shown in Figure 4.6. Again, part a shows that HARP is accurate at various dataset dimensionalities, and part b shows that the execution time was sub-quadratic with respect to d. It should be noted that most existing projected clustering algorithms would find difficulty in clustering these datasets since the dataset dimensionality is large and thus the number of relevant dimensions of each cluster is hardly predictable. Figure 4.6c shows the results on the dataset with 500 dimensions, speeding up by using fewer threshold levels. The execution time was greatly reduced, but the clustering accuracy remained excellent.

4.2.2 Results on Real Data

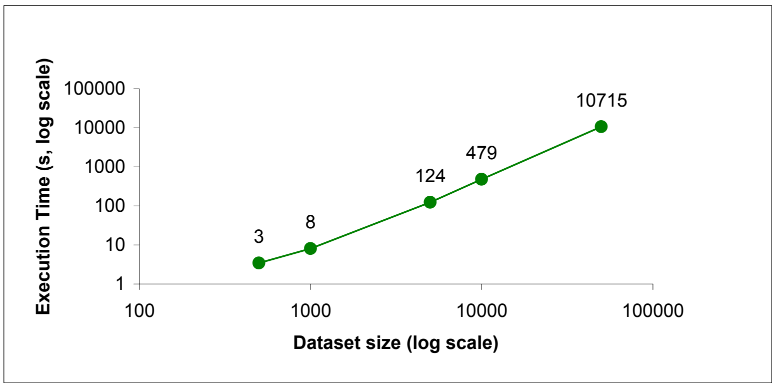
4.2.2.1 Lymphoma

We now present the experimental results on the real datasets. For the lymphoma data, we used HARP and PROCLUS as the representatives of projected clustering algorithms. The results with the best ARI values of each algorithm are shown in Table 4.4.

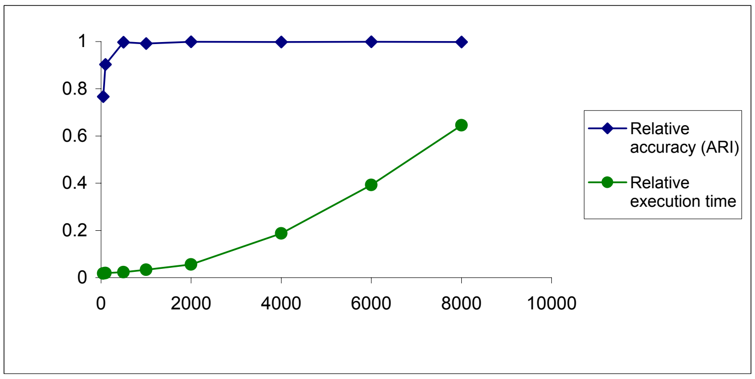
HARP got the best ARI score, even its clustering process was not guided by user parameters. We investigated the importance of dimension selection in



(a) Clustering accuracy of HARP with increasing N.

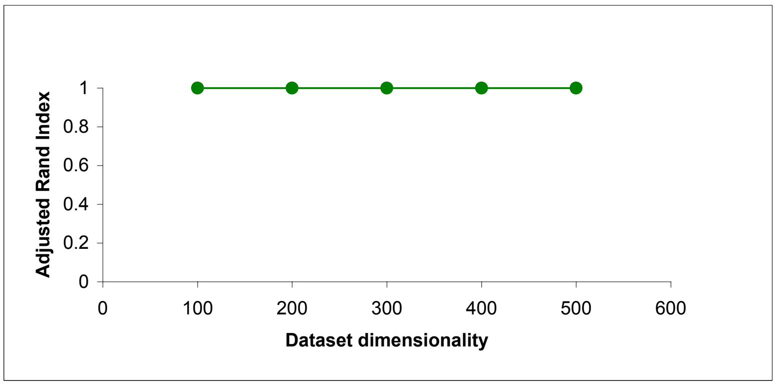


(b) Execution time of HARP with increasing N.

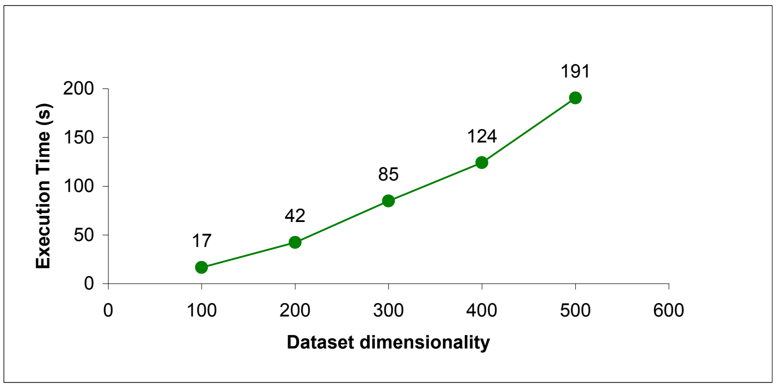


(c) Relative accuracy and execution time of HARP using various sample size on the dataset with $N = 10000$.

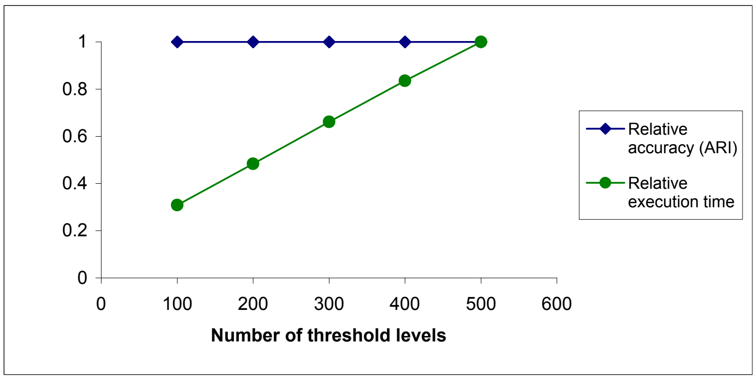
Figure 4.5. Clustering results of HARP with various dataset sizes.



(a) Clustering accuracy of HARP with increasing d.



(b) Execution time of HARP with increasing d.



(c) Relative accuracy and execution time of HARP using various number of threshold levels on the dataset with $d = 500$.

Figure 4.6. Clustering results of HARP with various dataset dimensionalities.

Algorithm	Best ARI
HARP	0.75
PROCLUS	0.64
KPrototype	0.63
CLARANS	0.61
Hierarchical	0.49
CAST	0.48

Table 4.4. The best ARI values achieved by various algorithms on the lymphoma data.

the formation of the clusters by calculating the distance ratios A_1 to A_3 of them. Table 4.5 lists the ratios of some interesting clusters located at the top two levels of the dendrogram. All of them satisfy the three requirements listed in Section 4.1.5, which means the selection of relevant dimensions makes the cluster members more distinguishable. For each cluster of samples, we also randomly selected 100,000 sets of relevant dimensions and calculated the corresponding distance ratios. All the resulting ratios are very close to one with standard deviations not more than 10^{-5}, which verify that the relevant dimensions selected by HARP are statistically unexpected and significantly better than random selections.

The results in Table 4.4 also reveal that projected clustering algorithms (HARP and PROCLUS) performed better than non-projected algorithms on this dataset, but the performance difference is not prominent. This may be explained by the large numbers of selected genes of the clusters listed in Table 4.5, which range from 61% to 90% of the dataset dimensionality. According to our previous results shown in Figure 4.1, projected clustering algorithms only have moderate advantage over non-projected algorithms in this range of l^r values.

We then examined the biological meaning of the relevant dimensions of the clusters. In Figure 2 of [8], some genes are highlighted as the signatures of some sample types or biological processes. The genes are divided into four regions:

Samples	No. of selected genes	A_1	A_2	A_3
6 RAT	2456	0.72	1.32	0.87
43 DLBCL, 2 NILNT	3515	0.96	1.25	1.02
10 ABB, 1 TCL	2734	0.80	1.32	1.00
9 FL, 2 GCB, 2 RBB	3104	0.85	1.38	1.00
11 CLL, 2 RBB	2614	0.82	1.27	0.97
16 DLBCL	3347	0.90	1.38	1.01
27 DLBCL, 2 NILNT	3610	0.96	1.32	1.00

Abbreviations: ABB=activated blood B, CLL=mantle cell lymphoma and chronic lymphocytic leukemia, DLBCL=diffuse large B-cell lymphoma, FL=follicular lymphoma, GCB=germinal centre B, RAT=resting/activated T, RBB=resting blood B, NILNT=NI. lymph node/tonsil, TCL=transformed cell lines.

Table 4.5. The distance ratios of some interesting clusters identified by HARP from the lymphoma data.

proliferation, germinal centre B, lymph node and T cell. For each cluster formed by HARP, we sorted all the genes in descending order according to their R values, and checked the ranks of the signature genes. It was found that the large DLBCL cluster has many signature genes in the proliferation region receiving high ranks, which suggests that the expression values of the genes could potentially be used to identify DLBCL samples. Similarly, it was found that the resting/activated T samples have a distinctive expression pattern. The 6 samples form a clear cluster with many of the signature genes receiving very large R values. Activated blood B, FL and CLL samples formed three separate clusters consisting of few samples from other types. They all have large R values at the signature genes at the lymph node region due to the constantly low expression, but the three types of samples were successfully separated into different clusters according to the expression values of other relevant genes, in particular those in the germinal centre B region. A complete list of the ranks and R values of the signature genes in different clusters can be found at `http://www.csis.hku.hk/~ylyip/harp/`.

4.2.2.2 Leukemia

In [32], 50 informative genes that have very different expression patterns in the two classes are used to build a highly accurate classifier. This suggests that a very small number of relevant genes are enough to distinguish the two types of samples. We therefore initialized d_{min} to 50 in order to select a small set of highly relevant genes for each cluster. Notice that unlike setting the l parameter of PROCLUS and ORCLUS, initializing d_{min} to a certain value does not force HARP to select any specific number of genes for each cluster. HARP is free to select any number of genes *not less than* d_{min}. The setting simply suggests HARP to focus on the genes with larger R values. With this setting, HARP produced one cluster that contained only ALL samples and the other contained mainly AML samples with only 3 errors (ARI: 0.71), which is a mild improvement over the clustering result presented in [32] (4 errors, ARI: 0.62). The ALL and AML clusters identified by HARP have 112 and 59 selected genes respectively, both with average R values of 0.95, which indicate the extremely high distinguishing power of the genes. By examining the dendrogram, we also found that the 8 T-ALL samples formed its own cluster before merging with any B-ALL samples. The pure T-ALL cluster has 151 selected dimensions with average R value of 0.99, which are potential signature genes for distinguishing T-ALL from the other two types of samples.

We then calculated the distance ratios A_1 to A_3 of the two final clusters and the T-ALL cluster (Table 4.6). Comparing the ratios with those of the lymphoma clusters (Table 4.5), the A_1 ratios are much lower and the A_3 ratios are much higher. This indicates that dimension selection is more beneficial to the leukemia dataset by making the clusters more compact and more distant from each other. In contrast, the A_2 ratios are just slightly larger than one since only a small amount of dimensions are selected for each cluster, which means object distances in the non-selected subspace are not much different from those calculated in the input space.

Samples	No. of relevant genes	A_1	A_2	A_3
16 B-ALL, 8 T-ALL	112	0.40	1.01	2.97
11 AML, 3 B-ALL	59	0.35	1.00	2.81
8 T-ALL	151	0.24	1.01	2.07

Table 4.6. The distance ratios of the two final clusters and the pure T-ALL cluster identified by HARP from the leukemia data.

Algorithm	Avg. no. of genes	Avg. no. of time points	Avg. H score	Avg. score to size ratio
Cheng and Church	167	12	204	0.10
HARP	243	10	203	0.08

Table 4.7. Comparison of the clusters identified by HARP and those reported in [18] from the yeast data.

4.2.2.3 Yeast

Next we performed pattern-based clustering on the yeast dataset. We used HARP to produce about 100 distinct clusters and compared them with the 100 biclusters discovered in [18]. Table 4.7 compares some statistics of the two sets of clusters. On average the clusters produced by HARP contain more genes but fewer time points. They also have a slightly better average squared residue score to size (number of genes multiplied by number of time points) ratio. Figure 4.7 and Figure 4.8 show the clusters with the best absolute scores and score to size ratios. According to the results, HARP was able to identify clusters with diverse sizes and dimensionalities. It also successfully grouped together genes with similar expression patterns but in opposite directions. The average size of the clusters also suggests that a significant number of genes were assigned to multiple clusters with matched signatures.

We evaluated the biological significance of the clusters by a phenotypic cate-

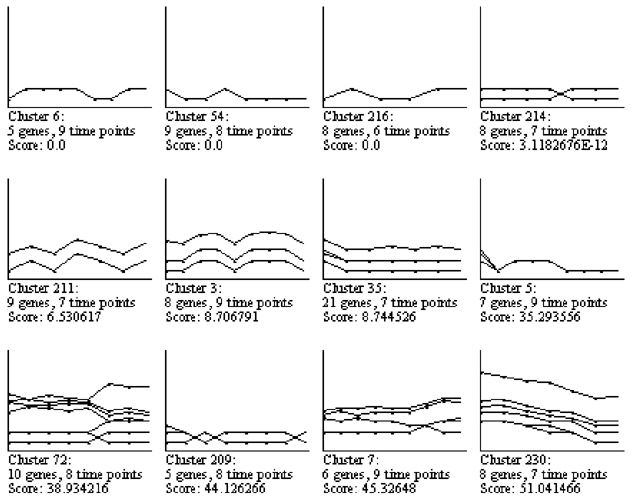


Figure 4.7. The clusters identified by HARP from the yeast data with the best mean squared residue scores.

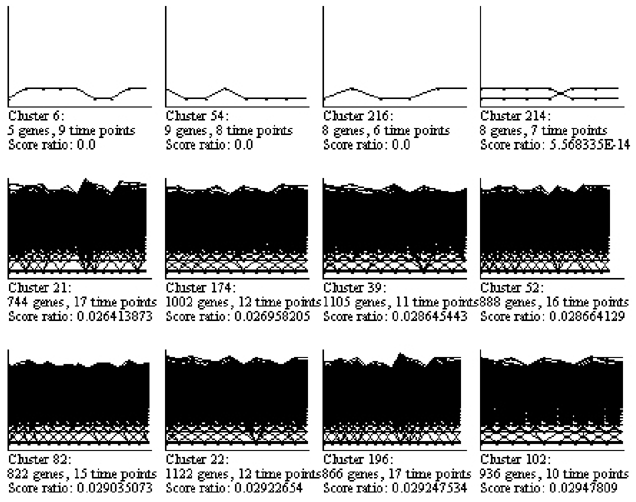


Figure 4.8. The clusters identified by HARP from the yeast data with the best mean squared residue score to size ratios.

Category: genes
Budding, directional growth: YDR507C
Cell cycle regulators: YPL256C, YJL187C
Chromosome, nuclear segregation: YMR076C, YDL003W, YKL042W, YMR078C
DNA repair and recombination: YLR383W, YDR097C
DNA replication: YOR074C, YLR103C, YAR007C, YNL312W, YDL164C, YBR088C

Table 4.8. One of the clusters (cluster 53, no. of genes=22) identified by HARP from the yeast data that contains a significant amount of genes from related categories (all in late G1 phase).

gorization of mRNAs that are regulated with the cell cycle[2]. Some clusters were found to contain a significant amount of genes from related categories. One such clusters is shown in Table 4.8, which contains many categorized genes in the late G1 phase, with functions ranging from budding, cell cycle regulation, nuclear segregation to DNA replication and repair.

4.2.2.4 Food

For the food data, we used HARP to produce twenty clusters. Some interesting clusters are summarized in Table 4.9. For example, one of them contains all twelve margarine items in the dataset, which strongly suggests that the cluster is meaningful. 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 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 may not be able to produce the same interesting cluster.

[2]http://yscdp.stanford.edu/yeast_cell_cycle/functional_categories.html

Members	Items	Selected dimensions (mean, R value)
Bread and cereal	93	Fat (-0.43, 0.98)
		Food Energy (0.48, 0.94)
		Cholesterol (-0.38, 1.00)
		Saturated Fat (-0.45, 0.99)
Cake and muffin	22	Fat (-0.04, 0.98)
		Food Energy (0.52, 0.96)
		Protein (-0.14, 0.95)
		Saturated Fat (-0.08, 0.98)
Vegetable oil	18	Fat (4.31, 1.00)
		Carbohydrate (-0.95, 1.00)
		Protein (-0.78, 1.00)
		Cholesterol (-0.22, 0.80)
Margarine and salad dressing	13	Carbohydrate (-0.95, 1.00)
		Protein (-0.75, 1.00)
		Cholesterol (-0.38, 1.00)
Salted and unsalted butter	6	Carbohydrate (-0.95, 1.00)
		Protein (-0.75, 1.00)
		Saturated Fat (7.06, 1.00)

Table 4.9. Some interesting clusters identified by HARP from the food data.

4.2.2.5 Other results

Besides the results presented above, we have also conducted experiments on a number of gene expression datasets from which the results are less encouraging. On one extreme, some of the datasets can easily be clustered by non-projected clustering algorithms, which means that similarity calculations in the full input space are effective enough to distinguish the members of each cluster without performing dimension selection. On the other extreme, the clusters in some datasets are hard to determine even by supervised learning algorithms. For example, in a study of central nervous system embryonal tumor [58], a dataset was generated that contains two types of medulloblastoma samples: classic and desmoplastic. HARP was unable to separate the two types of samples to different clusters. We analyzed the dataset and discovered that a large proportion of dimensions (genes) has very low R values in the real cluster of classic medulloblastomas. Among the 7129 dimensions, the R values of 5575 (78%) are negative. This means the samples of the classic group have very different expression patterns, which may suggest subgroups within the samples. The situation for the desmoplastic group is much better, with only 1445 (20%) of the dimensions having negative R values. Unfortunately, some of the desmoplastic samples are very similar to some classic samples, even more similar than to other desmoplastic samples. This hinders the formation of a pure desmoplastic cluster before merging with other clusters that contain classic samples, which, if being formed, could be observed from the dendrogram.

According to all the experimental results, we believe that projected clusters do exist in some gene expression datasets. More importantly, HARP outperforms non-projected algorithms on these datasets, while it has comparable performance on the datasets where projected clusters appear to be absent. This strongly supports the use of HARP in gene expression data analysis.

Chapter 5

Further Discussions and Future Works

The results on the gene expression datasets show that HARP is able to identify statistically and biologically meaningful clusters without the aid of user parameters. The algorithm is thus especially useful when there is a large number of datasets to analyze or when there is little knowledge about the datasets, when time-consuming parameter tuning is not possible. In such situations, HARP can be used to automatically identify some interesting clusters for later, more labor-intensive analysis.

The dynamic threshold loosening mechanism of HARP is shown to be successful in avoiding the introduction of user parameters. Although the current loosening scheme is only a simple synchronous linear interpolation of the two internal thresholds, the results are already quite encouraging. We believe the concept of dynamic parameter tuning has a great potential in projected clustering and other problems to which the solutions usually rely on some user parameters.

The experimental results on synthetic data suggest that projected clustering has a pronounced advantage over non-projected clustering 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 average cluster dimensionalities not more than 30% of d. In some gene expression data, the number of relevant genes of each function group can be lower than 10% of the total number of genes involved in the experiments. As shown in Figures 4.1a, and 4.1b, most projected clustering algorithms are unable to correctly cluster datasets with l^r much smaller than d. HARP is relatively superior in such situations, but its performance can also get worse when l^r is as low as 10% of d. One way to deal with the problem is to initialize d_{min} to a small value, as what we did when working on the leukemia data. While this resulted in some nice results in this particular case, the solution is feasible only when there are some knowledge of the suitable initialization value (or range of values) of d_{min}.

On the other hand, the bottom-up searching approaches described in Section 2.1.1 are designed for clusters of low dimensionality, which seem to be more suitable for analyzing such datasets. Unfortunately, while the ratio $\frac{l^r}{d}$ is low, the absolute value of l^r can still be too high for these algorithms to handle due to their exponential time complexity with respect to cluster dimensionality. For instance, if 10% of genes are relevant to a cluster of samples in a small dataset that records the expression values of only 2000 genes, the dimensionality of the cluster would be 200. This means dense regions could be found in $2^{200} - 1$ non-empty subspaces, which is intractable for the current subspace clustering algorithms. Further improvements of projected clustering algorithms are called for.

One possible way is to use a small amount of external inputs to greatly reduce the size of the search space. When working on gene expression datasets, we noticed that it is common to find some domain knowledge that can guide the clustering process, but the knowledge is not sufficient for supervised learning. For example, a general problem of hierarchical clustering algorithms is that towards the end of clustering, some clusters containing objects from different real clusters could be forced to merge together due to the presence of small outlier clusters. The resulting clustering accuracy could drop substantially due to the incorrect merge. This unfavorable situation could be avoided if a little domain knowledge is applied to disallow the merging of the clusters by identifying that they contain objects known to come from different real clusters. The domain knowledge can be applied directly by the users, or extracted from the rich external information sources such as sequence (e.g. GenBank [14]), annotation (e.g. Gene Ontology [11]) and literature (e.g. PubMed [1]) databases. In the computer science community, some works have been started on this semi-supervised clustering problem (see for example [46, 67]). We believe the application of this technique in gene expression data analysis will soon become a hot research topic.

As mentioned in Section 1.3, feature selection techniques alone cannot solve the projected clustering problem since they do not determine a separate subspace for each cluster. However, when a dataset contains some noise dimensions that are irrelevant to all clusters, or when the clusters are not axis-parallel, dimension reduction methods such as principal component analysis (PCA) or independent component analysis (ICA) can be useful in filtering and transforming the data for projected clustering. There have been some studies on the effectiveness of such techniques on gene expression data [50, 70]. A future work of the current study is to compare the clustering results of projected and non-projected algorithms on gene expression data with and without performing such data preprocessing.

One difficulty that we have encountered during the study is to develop a formal procedure for evaluating the statistical significance of projected clusters. Functions developed for internal validation of non-projected clusters that require the generation of random clusters (e.g. U-statistic [42]) become computationally infeasible in the projected case. This is because in order to gather sufficient data of a cluster for statistical calculations, not only should the generated clusters have the same size as the target cluster, they should also share the same number of selected dimensions with it. As a result an extraordinarily large number of clusters have to be generated, which takes a huge amount of time, especially as projected clustering algorithms are generally less efficient than their non-projected counterparts.

In fact, evaluation of clustering results and the robustness of algorithms has become an important topic of bioinformatics research due to the vast amount of clustering algorithms available and the lack of clear guidelines on which algorithms to use under different situations. There are some recent studies on the topic based on the consistency of algorithms when different data attributes are deleted [21, 71], but as far as we know no similar methods have been proposed for projected clustering. We leave this topic as a future extension of the current study.

Likewise, we have encountered difficulties when trying to develop an objective score for projected clustering that is not biased by cluster dimensionality. We have tried to tailor the Davies-Bouldin Index (DBI) [22] to take care of dimension selection. We believed the DBI is more robust than the W^p score since it also considers the separation between different clusters. While we have successfully proved that the resulting function does not have the monotonically non-increasing property of the W^p score (Section 2.2), i.e., its value may become worse by deselecting some selected dimensions, empirical results show that it is still biased by cluster dimensionality, although not as severe as the W^p score. We look forward to other proposals for the evaluation functions.

The object assignment extension produced some nice non-disjoint clusters on the yeast dataset, but in general some important clusters can be missed if their structures are not captured by some disjoint clusters before object assignment. We propose two future extensions of HARP for identifying these clusters: to allow each cluster to be merged with multiple clusters, and to produce disjoint clusters on different small data samples, and then reassign other objects to the clusters. Both approaches allow the discovery of more projected structures.

The quality of the yeast clusters produced by HARP is comparable to those produced by the Cheng and Church algorithms, which were designed to optimize the pattern-based objective score. This suggests that non-projected clustering methods that assume distance-based object similarity can also be used in pattern-based clustering. Actually, in non-projected clustering, it can be easily proved that by standardizing a dataset such that each row has zero sum and unit sum of squares, the Euclidean distance between two objects in the transformed data is equal to $2 - 2r$, where r is the Pearson correlation between the objects in the original data [9]. This means the Euclidean distance between two objects in the transformed data reflects the dissimilarity between the rise and fall patterns of the objects in the original data. The pattern-based clustering problem is thus transformed to a distance-based clustering problem by the standardization process. The situation is more complicated in the projected case in that each cluster has its own set of relevant dimensions. As discussed in Section 3.6, standardization should be performed based on the projected values on such dimensions only. The trickiest thing is that the real relevant dimensions are unknown when standardization is performed. It becomes even more complicated when clusters are non-disjoint, at which a single projected value is subject to the standardization process of all the clusters that it is involved. We leave the more advanced methods of adaptive subspace standardization as a future work on the topic.

Using the speedup methods, typical gene expression datasets could be analyzed by HARP very efficiently. However, the adaptive readjustment of expression values in the pattern-based clustering extension requires heavy computations, which greatly lowers the efficiency of HARP. We will look for methods that can efficiently perform adaptive readjustment in further studies. Also, after each threshold loosening, $O(N^2)$ merge score calculations are required due to the change of the two thresholds. We will try to modify the definition of the merge score function such that components calculated in previous threshold levels can be reused to potentially save a substantial amount of time spending on merge score calculations.

All the algorithms considered in this thesis are memory-residence. As more and more microarray experiments are performed and the density of spots on each DNA chip becomes higher and higher, the size of gene expression datasets may soon become too large to be analyzed fully in main memory. A lot of efforts have been paid on storing gene expression profiles in databases, but there are few studies on the development of disc-based projected clustering algorithms. The need would probably become apparent in the near future.

Another issue related to the speed performance of clustering algorithms is whether they can be parallelized. There have been extensive studies on parallelized traditional clustering algorithms [49, 56]. It can be easily observed that there are plenty of rooms to improve the speed performance of HARP by parallelization. For instance, the $O(n^2)$ similarity scores in each threshold level can well be computed in parallel by different machines. At the current stage, most works on gene expression data clustering are concentrated on the quality of the results. Most projected algorithms are under rigorous improvements by new algorithms. When some existing algorithms become stable and more widely accepted, the focus may shift to the speed performance of the algorithms, which may lead to more works on the parallelization of projected clustering algorithms.

In this thesis, as in most studies on projected clustering, datasets are assumed to contain numeric values only. This is a valid assumption in the current study since most gene expression datasets are numerical. In some other applications, datasets may contain categorical attributes. The signatures of a cluster then resemble a rule in a decision tree [60], but they are discovered without using the information of class labels. In a preliminary study, we modified HARP to handle categorical data by replacing the relevance index R_{Ij} by the formula $1 - \frac{1-LocalModeRatio_{Ij}}{1-GlobalModeRatio_{Ij}}$, where $LocalModeRatio_{Ij}$ is the ratio of objects in cluster C_I having the mode value of attribute v_j of the cluster, and $GlobalModeRatio_{Ij}$ is the ratio of objects in the whole dataset having that value. The R^*_{Ij} score of a new cluster is redefined as the average of the R_{Ij} scores of the two merging clusters. As in the numeric case, R_{Ij} measures how similar are the members of cluster C_I at attribute v_j and how unlikely the similarity is due to the abundance of the attribute value globally. A new cluster has a high R^*_{Ij} score if and only if the two merging clusters both have a high R_{Ij} score, and their mode values at attribute v_j are the same. Essentially, the categorical version of the two functions captures the original ideas of the numerical version. Some initial experimental results show that the resulting algorithm has a performance close to the state-of-the-art categorical clustering algorithm ROCK [34] on some synthetic datasets as well as two real datasets Voting and Mushroom from the UCI machine learning repository [2]. We will look for some real categorical datasets that contain subspace clusters to extend the study of projected clustering on categorical data.

Chapter 6

Conclusions

In this thesis, we reviewed the various problems of finding clusters in subspaces and some proposed approaches to the problems in the literature. In particular, we analyzed the major challenges of the projected clustering problem, and suggested the reasons for the dependence of some existing projected clustering algorithms on user parameters. Based on the analysis, we proposed a new projected clustering algorithm HARP that does not depend on user inputs in determining the relevant dimensions of clusters, which makes it practical for applications where the correct values of the parameters are unknown. HARP makes use of the relevance index, histogram-based validation and dynamic threshold loosening to dynamically adjust the merging requirements of clusters according to the current clustering status. It can also be extended to perform pattern-based clustering and produce non-disjoint clusters by adaptive mean-centering and post-clustering object assignment respectively. The experimental results on synthetic data suggest that HARP has a higher accuracy and usability than the projected and non-projected algorithms being compared, and it remains highly accurate on noisy datasets and datasets that contain imperfect clusters. The experimental results on real datasets show that HARP works well in situations where object similarity is based on either distance or expression pattern, and where disjoint or non-disjoint clusters are required. The clusters identified are both statistically and biologically meaningful. We also discussed the limitations and some possible future research directions of HARP and other projected clustering algorithms.

To increase the utility and evaluation of HARP, we are in the process of establishing a seamless interoperation between the Yale Microarray Database [19] and HARP through the use of XML-based web services. This allows the users of YMD to extract data from the microarray experiments of their interest and send the data directly to the remote HARP web service for cluster analysis.

Appendix A

How Likely is a Cluster Correct?

Consider two clusters C^c and C^i of equal size n, where the objects in C^c come from the same real cluster with d_I relevant dimensions, while the objects in C^i are randomly sampled from the dataset. All dimensions are regarded as irrelevant to C^i. Suppose the relevance threshold R_{min} is fixed at a value such that the probabilities for each real relevant dimension and each real irrelevant dimension to be selected are p and q respectively. Assume $p = \gamma q$, where $\gamma \geq 1$. Now, given a cluster C chosen from the two clusters, we want to determine how likely C is in fact C^c in comparison to C^i if C has l selected dimensions.

Denote P(c) and P(i) be the probability that C is C^c and C^i respectively, and P(l) be the probability that a cluster of n objects has l selected dimensions. By Bayes' theorem,

$$\begin{aligned} P(c|l) &= \frac{P(l|c)P(c)}{P(l)} \\ &= \frac{P(l|c)P(c)}{P(l|c)P(c) + P(l|i)P(i)}. \end{aligned} \tag{A.1}$$

Similarly,

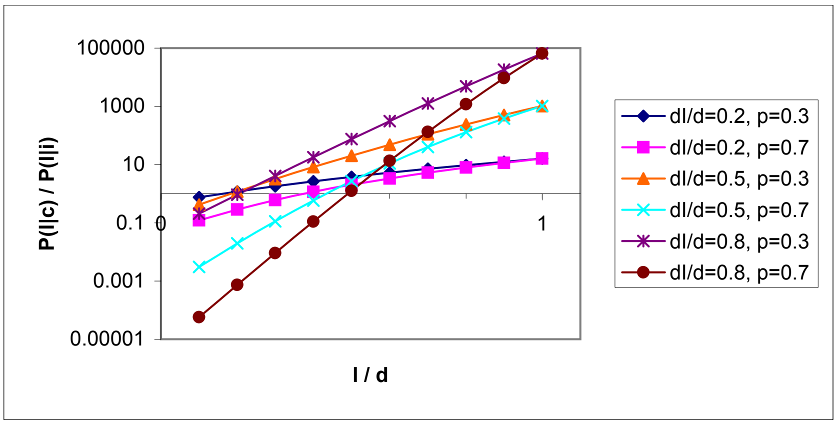
$$P(i|l) \quad = \quad \frac{P(l|i)P(i)}{P(l|c)P(c) + P(l|i)P(i)}. \tag{A.2}$$

Dividing A.1 by A.2, we get the relative probability for C to be C^c over C^i:

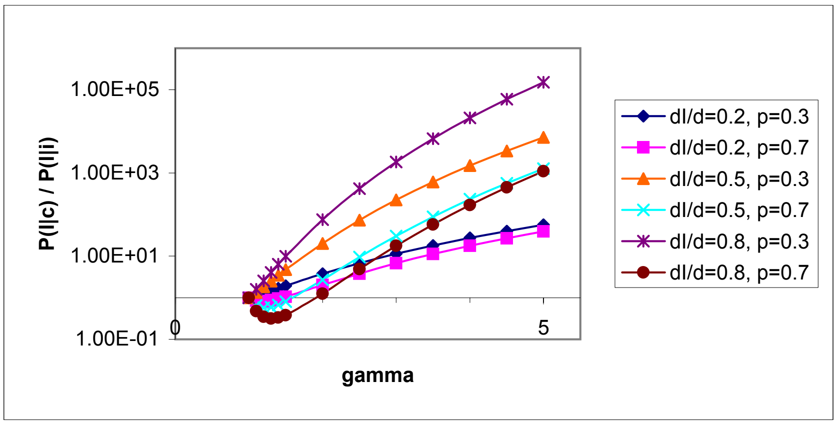
$$\begin{aligned}
\frac{P(c|l)}{P(i|l)} &= \frac{P(l|c)P(c)}{P(l|i)P(i)} \\
&\propto \frac{P(l|c)}{P(l|i)} \\
&= \frac{\sum_{0 \le r \le l \wedge l+d_I-d \le r \le d_I} \binom{d_I}{r} p^r (1-p)^{d_I-r} \binom{d-d_I}{l-r} q^{l-r} (1-q)^{d-d_I-l+r}}{\binom{d}{l} q^l (1-q)^{d-l}} \\
&= \frac{1}{\binom{d}{l}} \sum_{0 \le r \le l \wedge l+d_I-d \le r \le d_I} \binom{d_I}{r} \binom{d-d_I}{l-r} \gamma^r \left(\frac{1-\gamma q}{1-q}\right)^{d_I-r}, \qquad \text{(A.3)}
\end{aligned}$$

where r is the number of real relevant dimensions of C^c being selected. Figure A.1 shows the plot of the relative probability against different l/d and γ values. Six curves are shown in each figure, which correspond to different d_I and p value combinations. In Figure A.1a, the relative probability is shown to be monotonically increasing as l/d increases, which means the probability for a cluster to be correct is always higher if it has more selected dimensions. When $l = d$, the relative probability reaches its maximum value of γ^{d_I}. The figure also shows that as l/d decreases, the relative probability has a sharper drop along the curves with higher p values. This is because when p is large, it is very unlikely that a large number of relevant dimensions are not being selected.

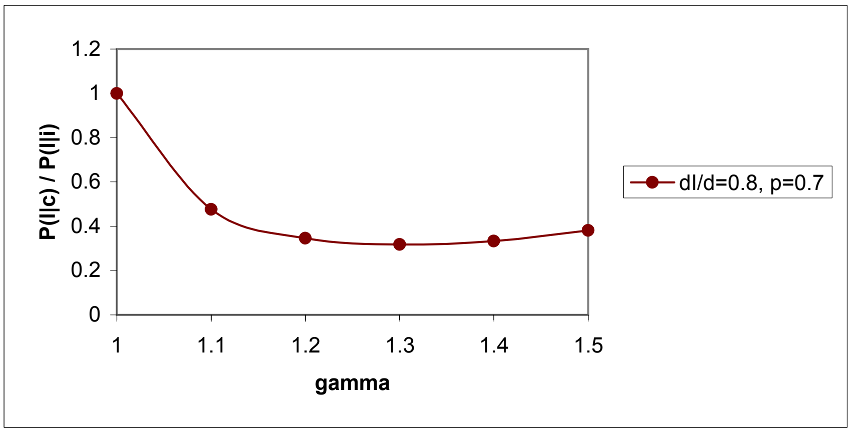
Figure A.1b shows that the relative probability has a general increasing trend as γ increases except when γ is small and p is large. Suppose the projected values of a relevant dimension and an irrelevant dimension are generated according to a Gaussian distribution and a uniform distribution respectively, it can be shown that γ increases monotonically as R_{min} increases at large R_{min} values. This means when R_{min} is sufficiently large, the relative probability is generally higher when R_{min} is larger. The abnormal declination of the relative probability when



(a) Changing $l/dratio$ ($\gamma = 2$).



(b) Changing γ ($l/d = 0.5$).



(c) A magnification of the region $1 \leq \gamma \leq 1.5$.

Figure A.1. A plot of the relative probability against different l/d and γ values ($d = 20$).

γ is small and p is large is explained by Figure A.1c, which is a magnification of Figure A.1b at the region $\gamma \in [1, 1.5]$. The nadir occurs when $\gamma = 1.3$, which is close 1.4, at which l/d $(= 0.5)$ equals the expected portion of selected dimensions in C^i $(q = p/\gamma = 0.5)$. Preceding to the minimum point, l is smaller than the expected number of selected dimensions of C^i, especially when p, and thus q, is large. Increasing γ effectively lowers the expected number, which makes C more likely to be C^i. The effect on C^c is much smaller as its expected number of selected dimensions is greatly determined by p, which remains unchanged. Beyond the minimum point, further increasing γ will make C to have more selected dimensions than the expected number of C^i, which makes C less likely to be C^i and thus the relative probability to increase.

In general, therefore, a cluster is more likely to contain objects from the same real cluster if it has more selected dimensions and each selected dimension is required to have a larger R value, which justifies the use of the dynamic threshold loosening mechanism.

Appendix B

List of Symbols

D	The working dataset
N	The size of (number of objects in) D
V	The set of all dimensions of D
d	The dimensionality of D
C_I	The I-th cluster
N_I	The size of (number of objects in) C_I
V_I	The set of relevant dimensions of C_I
d_I	The dimensionality (number of relevant dimensions) of C_I
$W(\{C_I\})$	The average within-cluster distance to centroid of the set $\{C_I\}$ of clusters
W_I	The average within-cluster distance to centroid of cluster C_I
k	The number of clusters formed, or the target number of clusters to form
v_j	The j-th dimension

x	An object
x_j	The projected value of object x on v_j
x_{Ij}	The average projected value of all objects in C_I on v_j
σ^2_{Ij}	The variance of projected values of all objects in C_I along v_j
$W^p(\{C_I\})$	The projected version of $W(\{C_I\})$
$W^p_I(V_I)$	The projected version of W_I with the relevant dimensions specified by V_I
ω	The width parameter of the DOC, FastDOC and MineClus algorithms
$\mu(N_I, d_I)$	The cluster evaluation function of the DOC, FastDOC and MineClus algorithms
β	A user parameter used in $\mu(N_I, d_I)$ to define the relative importance of the size and dimensionality of a cluster
l	A user parameter of the PROCLUS, ORCLUS and OPSM algorithms
H_I	The mean squared residue score of C_I
x_{iJ}	The average projected value of object x_i on the relevant dimensions of C_I
x_{IJ}	The average of the projected values of all objects in C_I on all relevant dimensions of C_I
δ	The quality threshold used in the Cheng and Church algorithms and the pCluster model
s	The cluster dimensionality parameter of the OPSM algorithm

G	The pool of gene sets in the coupled two-way clustering approach
S	The pool of sample sets in the coupled two-way clustering approach
$\sigma^2_{\cdot j}$	The variance of projected values of all objects in D along v_j
R_{Ij}	The relevance index of dimension v_j in cluster C_I
Q_I	The quality score of cluster C_I
R^*_{Ij}	The mutual-disagreement-sensitive relevance index of dimension v_j in cluster C_I
$R_{I_1j\|I_2}$	The adjusted relevance index of v_j in C_{I_1} given that C_{I_1} is merging with C_{I_2}
$MS(C_{I_1}, C_{I_2})$	The merge score between clusters C_{I_1} and C_{I_2}
min_{Ij}	The minimum projected value of the members of C_I on v_j
max_{Ij}	The maximum projected value of the members of C_I on v_j
d_{min}	The dimensionality threshold of HARP
R_{min}	The relevance threshold of HARP
l^r	The average dimensionality of the real clusters in testing data
min_j	The minimum projected value of all objects in D on v_j
max_j	The maximum projected value of all objects in D on v_j
e	Artificial data error rate in synthetic data
o	Artificial outlier rate in synthetic data
U^r	The partitioning of objects according to the real clusters
U^c	The partitioning of objects in a clustering result

$A_1(C_I)$	Average selected-to-all within-cluster distance to centroid ratio of cluster C_I
$A_2(C_I)$	Average non-selected-to-all within-cluster distance to centroid ratio of cluster C_I
$A_3(C_I)$	Average selected-to-all between-cluster distance ratio of cluster C_I

Bibliography

[1] Pubmed. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi.

[2] Uci machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[3] Charu C. Aggarwal, Cecilia Procopiuc, Joel L. Wolf, Philip S. Yu, and Jong Soo Park. Fast algorithms for projected clustering. In *ACM SIGMOD International Conference on Management of Data*, 1999.

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

[5] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD International Conference on Management of Data*, 1998.

[6] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 1993.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Eleventh International Conference on Data Engineering*, 1995.

[8] Ash A. Alizadeh, Michael B. Eisen, R. Eric Davis, Chi Ma, Izidore S. Lossos, Andeas Rosenwald, Jennifer C. Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, John I. Powell, Liming Yang, Gerald E. Marti, Troy Moore, James Hudson, Lisheng Lu, David B. Lewis, Robert Tibshirani, Gavin Sherlock, Wing C. Chan, Timothy C. Greiner, Dennis D.Weisenburger, James O. Armitage, Roger Warnke, Ronald Levy, Wyndham Wilson, Michael R. Grever, John C. Byrd, David Botstein, Patrick O. Brown, and Louis M. Staudt. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.

[9] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA*, 96:6745–6750, 1999.

[10] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jrg Sander. OPTICS: Ordering points to identify the clustering structure. In *ACM SIGMOD International Conference on Management of Data*, 1999.

[11] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[12] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *Proceedings of the Sixth Annual International Conference on Computational Biology*, 2002.

[13] Amir Ben-Dor and Zohar Yakhini. Clustering gene expression patterns. In *Proceedings of the Annual International Conference on Computational Molecular Biology*, 1999.

[14] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and David L. Wheeler. Genbank. *Nucleic Acids Research*, 31(1):23–27, 2003.

[15] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science*, 1540:217–235, 1999.

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

[17] Chun Hung Cheng, Ada Wai-Chee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[18] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 2000.

[19] K. H. Cheung, K. White, J. Hager, M. Gerstein, V. Reinke, K. Nelson, P. Masiar, P. Srivastava, Y. Li, J. Li, J. M. Li, D. B. Allison, M. Snyder, P. Miller, and K. Williams. YMD: A microarray database for large-scale gene expression analysis. In *The American Medical Informatics Association 2002 Annual Symposium*, 2002.

[20] Raymond J. Cho, Michael J. Campbell, Elizabeth A. Winzeler, Lars Steinmetz, Andrew Conway, Lisa Wodicka, Tyra G. Wolfsberg, Andrei E. Gabrielian, David Landsman, David J. Lockhart, and Ronald W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73, 1998.

[21] Susmita Datta and Somnath Datta. Comparisons and validation of statistical clustering techniques for microarray gene expression data. *Bioinformatics*, 19(4):459–466, 2003.

[22] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2), 1979.

[23] Doulaye Dembl and Philippe Kastner. Fuzzy C-means method for clustering microarray data. *BioInformatics*, 19(8):973–980, 2003.

[24] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[25] Sandrine Dudoit, Robert Gentleman, Rafael Irizarry, and Yee Hwa Yang. Introduction to dna microarray technologies. http://www.bioconductor.org/workshops/WyethCourse101702/MarrayTech/MarrayTech4.pdf.

[26] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868, 1998.

[27] David Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[28] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1996.

[29] Brian S. Everitt and Graham Dunn. *Applied Multivariate Data Analysis (Second Edition).* Arnold, 2001.

[30] Audrey P. Gasch, Paul T. Spellman, Camilla M. Kao, Orna Carmel-Harel, Michael B. Eisen, Gisela Storz, David Botstein, and Patrick O. Brown.

Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11:4241–4257, 2000.

[31] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci. USA*, 97:12079–12084, 2000.

[32] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[33] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. In *ACM SIGMOD International Conference on Management of Data*, 1998.

[34] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. In *15th International Conference on Data Engineering*, 1999.

[35] Eui-Hong Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. Clustering based on association rule hypergraphs. In *1997 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[36] Jiawei Han and Micheline Kamber. *Dara Mining: Concepts and Techniques.* Morgan Kaufmann, 2000.

[37] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

[38] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28, 1979.

[39] Trevor Hastie, Robert Tibshirani, Michael B Eisen, Ash Alizadeh, Ronald Levy, Louis Staudt, Wing C Chan, David Botstein, and Patrick Brown.

‘Gene shaving’ as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1(2):research0003.1–0003.2, 2000.

[40] Javier Herrero, Alfonso Valencia, and Joaquin Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *BioInformatics*, 17(2):126–136, 2001.

[41] 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.

[42] Zhexue Huang, David W. Cheung, and Michael K. Ng. An empirical study on the visual cluster validation method with fastmap. In *Proceedings of the Ninth International Conference on Database Systems for Advanced Applications*, 2001.

[43] Vishwanath R. Iyer, Michael B. Eisen, Douglas T. Ross, Greg Schuler, Troy Moore, Jeffrey C. F. Lee, Jeffrey M. Trent, Louis M. Staudt, James Hudson Jr., Mark S. Boguski, Daval Lashkari, Dari Shalon, David Botstein, and Patrick O. Brown. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283:83–87, 1999.

[44] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, 1994.

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

[46] Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[47] Yuval Kluger, Ronen Basri, Joseph T. Chang, and Mark Gerstein. Spectral biclustering of microarray cancer data: Co-clustering genes and conditions. *Genome Research*, 13(4):703–716, 2003.

[48] Laura Lazzeroni and Art Owen. Plaid models for gene expression data. *Statistica Sinica*, 12:61–86, 2002.

[49] Xiaobo Li and Zhi Xi Fang. Parallel clustering algorithms. *Parallel Computing*, 11(3):275–290, 1989.

[50] Wolfram Liebermeister. Linear modes of gene expression determined by independent component analysis. *BioInformatics*, 18(1):51–60, 2002.

[51] Alexander V. Lukashin and Rainer Fuchs. Analysis of temporal gene expression profiles: Clustering by simulated annealing and determining the optimal number of clusters. *Bioinformatics*, 17(5):405–414, 2001.

[52] Daniela Matei, Thomas G Graeber, Rae Lynn Baldwin, Beth Y Karlan, Jianyu Rao, and David D Chang. Gene expression in epithelial ovarian carcinoma. *Oncogene*, 21:6289–6298, 2002.

[53] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[54] H. Nagesh, S. Goil, and A. Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. Technical Report 9906-010, Northwestern University, June 1999.

[55] Raymond T. Ng and Jiawei 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.

[56] C. F. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21(8):1313–1325, 1995.

[57] Charles M. Perou, Therese Srlie, Michael B. Eisen, Matt van de Rijn, Stefanie S. Jeffrey, Christian A. Rees, Jonathan R. Pollack, Douglas T. Ross,

Hilde Johnsen, Lars A. Akslen, ystein Fluge, Alexander Pergamenschikov, Cheryl Williams, Shirley X. Zhu, Per E. Lnning, Anne-Lise Brresen-Dale, Patrick O. Brown, and David Botstein. Molecular portraits of human breast tumors. *Nature*, 406:747–752, 2000.

[58] Scott L. Pomeroy, Pablo Tamayo, Michelle Gaasenbeek, Lisa M. Sturla, Michael Angelo, Margaret E. McLaughlin, John Y. H. Kim, Liliana C. Goumnerovak, Peter M. Black, Ching Lau, Jeffrey C. Allen, David Zagzag, James M. Olson, Tom Curran, Cynthia Wetmore, Jaclyn A. Biegel, Tomaso Poggio, Shayan Mukherjee, Ryan Rifkin, Andrea Califano, Gustavo Stolovitzky, David N. Louis, Jill P. Mesirov, Eric S. Lander, and Todd R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415:436–442, 2002.

[59] Cecilia M. Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *ACM SIGMOD International Conference on Management of Data*, 2002.

[60] J. Ross Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, 1993.

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

[62] M. Schena, D. Shalon, R. w. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–470, 1995.

[63] Frank De Smet, Janick Mathys, Kathleen Marchal, Gert Thijs, Bart De Moor, and Yves Moreau. Adaptive quality-based clustering of gene expression profiles. *Bioinformatics*, 18(5):735–746, 2002.

[64] Pablo Tamayo, Donna Slonim, Jill Mesirov, Qing Zhu, Sutisak Kitareewan, Ethan Dmitrovsky, Eric S. Lander, and Todd R. Golub. Interpreting pat-

terns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA*, 96:2907–2912, 1999.

[65] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(Suppl. 1):S136–S144, 2002.

[66] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22:281–285, 1999.

[67] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[68] Haisun Wang, Wei Wang, Jiong Yang, and Philip S. Yu. Clustering by pattern similarity in large data sets. In *ACM SIGMOD International Conference on Management of Data*, 2002.

[69] Jiong Yang, Haixun Wang, Wei Wang, and Philip Yu. Enhanced biclustering on expression data. In *Proceedings of the IEEE Third Symposium on Bioinformatics and Bioengineering*, 2003.

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

[71] K. Y. Yeung, D. R. Haynor, and W. L. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17(4):309–318, 2001.

[72] Man Lung Yiu and Nikos Mamoulis. Frequent-pattern based iterative projected clustering. In *IEEE International Conference on Data Mining*, 2003.

[73] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In *ACM SIGMOD International Conference on Management of Data*, 1996.